



GenRad

STATION
MANUAL
D/642

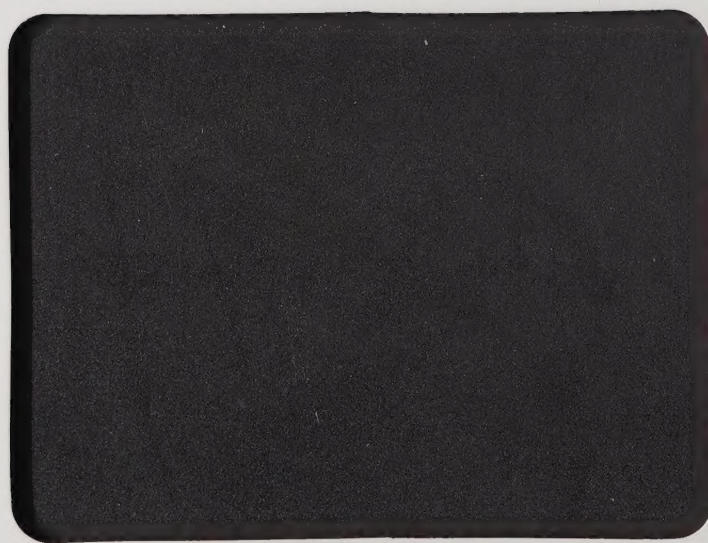
Test Programming

R Digital/Analog
Test Systems

Form 1796-0101-C
(RELEASE 03)

Instruction Manual

21000
MANUAL
D/e 45



Contents

Introduction — Section 1
Reference Material — Section 2
Description of Commands — Section 3
Program Considerations — Section 4
Sample Programs — Section 5
Instruction Summary — Appendix A
Executable PAL8 Techniques — Appendix B
CAPS-Related Techniques — Appendix C
GR 1795/1799 Digital and Power Supply Programming — Appendix D
IEC (ASCII) Bus Programming — Appendix E
GR 1796 Programming and 1799 Analog Programming — Appendix F
Index

Test Programming

GR Digital/Analog
Test Systems

Form 1796-0101-C
(RELEASE 03)

List of Effective Pages

INSERT LATEST CHANGED PAGES. DESTROY SUPERSEDED PAGES.

NOTE: The portion of the text affected by the changes is indicated by a vertical line in the outer margin of the page.

TOTAL NUMBER OF PAGES IN THIS PUBLICATION IS 216 CONSISTING OF THE FOLLOWING:

Issue

Original

Date

1 February, 1978

Page No.

Issue

Title	Original
A	Original
iii	Original
iv	Original
DD-1 through DD-3	Original
1-1 through 1-5	Original
2-1 through 2-10	Original
3-1 through 3-71	Original
4-1 through 4-3	Original
5-1 through 5-8	Original
A-1 through A-5	Original
B-1 through B-16	Original
C-1 through C-8	Original
D-1 through D-5	Original
E-1 through E-10	Original
F-1 through F-56	Original
Index I-1 through I-5	Original

*The asterisk indicates pages changed, added, or deleted by the current change.

Generalized equipment specifications are used throughout this manual to provide perspective and to aid the presentation of the subject matter. These specifications are of a general nature and should not be considered applicable to a particular test system.

The material in this manual is for informational purposes only and is subject to change without notice. GenRad assumes no responsibility for any errors or for consequential damages that may result from the use or misinterpretation of any of the procedures in this publication.

WARNING – SHOCK HAZARD

- Dangerous voltages may be present on test system shelf device connectors when SYSTEM POWER is ON.
- Systems equipped with the high compliance current source (DCVIF) must have device shelf pins connected to a load prior to use. Voltages as high as 200 V can be developed across open-circuited shelf pins assigned to DCVIF during test operations. Devices tested with this function should also be shielded from the operator.

Contents

SECTION 1 – INTRODUCTION

1.1	INTRODUCTION	1-1
1.2	THE GR TEST SYSTEMS	1-1
1.3	SYSTEM MANUALS	1-2
1.4	THE TEST LANGUAGE	1-2
1.5	SYSTEM DESCRIPTION	1-2

SECTION 2 – REFERENCE MATERIAL

2.1	GENERAL	2-1
2.2	DEFINITIONS	2-1
2.3	LIMITS	2-1
2.4	NUMERICAL VALUES	2-2
2.5	COMMENTS	2-2
2.6	CHARACTER SET	2-2
2.7	RESERVED WORDS	2-3
2.8	BRANCH ADDRESSES AND LABELS	2-8
2.9	REGISTERS	2-9
2.9.1	Registers Available	2-9
2.9.2	Register Operations	2-10

SECTION 3 – DESCRIPTION OF COMMANDS

3.1	GENERAL	3-1
3.2	TEST EQUIPMENT COMMANDS	3-1
3.2.1	General	3-1
3.2.2	Test And Measurement Commands	3-2
3.2.3	Explanation Of Command Examples	3-2
3.2.4	Terminal Designations	3-2
3.2.5	Earmarking CAPS Stimuli (CAPS/RECORD)	3-4
3.2.6	Logic Drivers And Sensors (LGC)	3-4
3.2.7	Device Power Supplies (PS-i)	3-17
3.2.8	Sync Pulse Generation (SYNC)	3-26
3.2.9	Relay Drivers (RLY)	3-27
3.2.10	DC Voltmeter (DCV)	3-27
3.2.11	DC Current Meter (DCI)	3-28
3.2.12	AC Voltmeter (ACV)	3-28
3.2.13	AC Source (ACS)	3-29
3.2.14	Frequency Measurement Unit (FMU)	3-30
3.2.15	Resistance Measurement Unit (RMU)	3-32
3.2.16	Pulse Generator (PGEN)	3-33
3.2.17	Time Measurement Unit (TMU)	3-35
3.2.18	Scanning System (S-SCAN, M-SCAN, P-SCAN)	3-33
3.2.19	External Test Equipment (XSOURCE,XMEAS)	3-35
3.2.20	Dc Voltage Measurement (DCVPRB) -Pulse-Catching Probe	3-39
3.3	ARITHMETIC COMMANDS	3-40
3.3.1	General	3-40
3.3.2	Load (LOAD or LD)	3-40
3.3.3	Store (STORE or STO)	3-40
3.3.4	Subtract (SUB)	3-40
3.3.5	Subtract (SUB)	3-36

3.3.6	Multiply (MPY or MULT)	3-41
3.3.7	Divide (DIV)	3-41
3.3.8	Square Root (SQRT)	3-41
3.3.9	Square (SQUARE)	3-41
3.3.10	Absolute (ABSOLUTE)	3-41
3.3.11	Arithmetic Test (ATEST)	3-41
3.3.12	Comparison (COMP)	3-41
3.3.13	Trigonometric (SIN, COS, ATAN)	3-42
3.3.14	Logarithmic (LOG, EXP)	3-42

3.4	UTILITY COMMANDS	3-42
3.4.1	General	3-42
3.4.2	No Operation (NOP)	3-43
3.4.3	Block (BLOCK)	3-43
3.4.4	End (END)	3-43
3.4.5	Branch (BRANCH)	3-43
3.4.6	Call (CALL)	3-44
3.4.7	Return (RETURN)	3-45
3.4.7A	Error Return (ERRET)	3-45
3.4.8	DO Loops (DO, DONE, ESCAPE)	3-46
3.4.9	Programmed Delays (DELAY)	3-46
3.4.10	Pause (PAUSE)	3-46
3.4.11	Clear All Test Equipment (CLEAR ALL)	3-47
3.4.12	Character Compare (CCOMP)	3-47
3.4.13	I/O Commands (READ,WRITE,OPEN, CLOSE)	3-47
3.4.14	Special Program (RESERVE,EXEC)	3-52
3.4.15	Symbolic Terminal (ASSIGN PNAME)	3-53
3.4.16	User Option Switches (USOPT1,USOPT2)	3-54
3.5	MACRO PROGRAMMING AID	3-55
3.5.1	Repeat/Loop Macros	3-55
3.5.2	System Macros (SEQUENCE,SHIFT, ROTATE,GRAY,RANDOM,IRANDOM)	3-57
3.5.3	User-Styled Macros (DEFINE,ENDM)	3-63

SECTION 4 – PROGRAM CONSIDERATIONS

4.1	GENERAL	4-1
4.2	PROGRAM CONSIDERATIONS	4-1

SECTION 5 – SAMPLE PROGRAMS 5-1

5.1	FIRST SAMPLE (Simple Test Language)	5-1
5.2	SECOND SAMPLE (User-Styled Macros)	5-3

APPENDIX A – INSTRUCTION SUMMARY A-1

APPENDIX B – EXECUTABLE PAL8 TECHNIQUES B-1

APPENDIX C – CAPS-RELATED TECHNIQUES C-1

APPENDIX D – GR 1795/1799 DIGITAL AND ANALOG PROGRAMMING D-1

APPENDIX E – IEC (ASCII) BUS PROGRAMMING E-1

APPENDIX F – GR 1796 PROGRAMMING AND GR 1799 ANALOG PROGRAMMING F-1

INDEX Index-1

For
Instruction Manual

1796/1799 TEST SYSTEMS

The 1796 test language is described in Appendix F. The language is generally compatible with test statements used to prepare programs for 1792 test systems with the following exceptions:

1. Port numbers (-n) for use with SASCAN-n, SBSCAN-n, MASCAN-n, and MBSCAN-n keywords differ between 1792 and 1796/1799 test systems. New 1796 port numbers and the actual pin or jack spare connections that result are listed in Table F-3. New 1799 port numbers and the actual pin or jack spare connections that result are listed in Table F-3A. The following table illustrates the overall differences between each test system when any of the mSCAN-n keywords are used:

Connects:		
Keyword	1792D	1796/1799
MASCAN-1	DCV (+)	ACV
MASCAN-2	ACV	FMU (A Channel)
MASCAN-3	FMU (A Channel)	Spare
MASCAN-4	RMU (P+)	Spare
MASCAN-5	Spare	DCV (+)
MASCAN-6	Spare	RMU (Guard)
MASCAN-7	N/A	RMU (V+)
MBSCAN-1	DCV (—)	Spare
MBSCAN-2	Spare	FMU
MBSCAN-3	FMU (B Channel)	Spare
MBSCAN-4	RMU (P—)	Spare
MBSCAN-5	Spare	DCV (—)
MBSCAN-6	Spare	RMU (Guard Sense)
MBSCAN-7	N/A	RMU (V—)
SASCAN-1	ACS-1	ACS-1
SASCAN-2	PGEN-2 (or Spare)	PGEN-2 (or spare)
SASCAN-3	RMU (+)	Spare
SASCAN-4	N/A	RMU (V+)
SASCAN-5	N/A	RMU (I+)
SBSCAN-1	PGEN-1	PGEN-1
SBSCAN-2	ACS-2 (or Spare)	ACS-2 (or Spare)
SBSCAN-3	RMU (I—)	Spare
SBSCAN-4	N/A	RMU (V—)
SBSCAN-5	N/A	RMU (I—)

2. The 1792 keywords XSOURCE-1, XMEAS-1, and XMEAS-2 may be used to connect external test equipment, however, actual pin or jack connections will differ between test systems. Appendix F describes both the 1796 and 1799 connections that are made by each keyword. The 1792 keyword XMEAS-3 **cannot** be used with 1796 or 1799 test systems.

3. Methods of opening and closing shelf scanner pins and MUX channel relays differ between test systems. While the 1792 scanner language can be used to connect 1796/1799 analog pins without major modifications, it is desirable to assign pins within the **same** group whenever possible to optimize scanner loading characteristics. The 1796 analog source and measurement shelf pins are assigned in groups of 12 while the 1796/1799 hybrid shelf pins are assigned in groups of 48 (see Figure F-6).

4. The keyword SHELF **must** be inserted in 1796/1799 scanner statements whenever it is desired to connect the ACS or PGEN to **unscanned** shelf pins. The keyword SHELF is not required in 1792 test systems.

The 1796 includes a series of **new** features (such as FAST high-speed digital testing, a 16-bit precision multimeter with two terminal guarded and unguarded measurement capability, and scanned hybrid pins that can be assigned as digital or analog pins) that are not available on 1792 test systems. Complete details are provided in Appendix F.

1799 TEST SYSTEMS

The 1799 is a medium priced hybrid test system with both digital and analog testing capability. Test language statements for the digital section and device power supplies are described in Appendix D. Test language statements for the analog section are described in Appendix F. The language is generally compatible with test statements used to prepare programs for 1796 test systems with the following exceptions:

Digital Section

1. FAST high-speed digital testing is not available.
2. Either fixed (internally adjustable) driver/sensors with a single Logic Level Set and adjustable high levels or fully programmable driver/sensors may be supplied. The programmable driver/sensor high and low drive and sense levels may be set by test program statements over a range of ± 15 V. Dual Logic Level Sets are also available for mixed logic testing.
3. When using the fully programmable driver/sensors, each group of 12 pins **must** be assigned to the **same** Logic Level Set.
4. A combination of fixed, hardware-adjustable Triple Power Supplies, and/or fully programmable Universal Power Supplies may be supplied for device testing with 1799 test systems. Power supply programming is described in Appendix D.
5. Power supply scanning is **not** available on 1799 test systems. Device power supplies are wired directly to a shelf connector.
6. A Time Measurement Unit (TMU) is **not** available on 1799 test systems. The unit may be supplied with 1792D or 1796 test systems.

Analog Section

1. Hybrid shelf scanner pins may be supplied in groups of 48 pins. Hybrid shelf pins can be assigned as digital driver/sensor pins and/or as analog pins that connect to analog source and measurement instruments. Dedicated **scanned** analog source and measure shelf pins are **not** supplied with 1799 test systems. All analog scanning is accomplished through the hybrid scanner. Previous test program scanner statements that do **not** reference hybrid shelf scanner pins through use of the C= or H= tokens described in Appendix F will have to be modified.
2. A **single** AC Source (ACS) and Pulse Generator (PGEN) are available on 1799 test systems. The software can support additional units.
3. External pins and jacks assigned through use of scanner language keywords (such as MASCAN-n, SHELF, etc.) differ between 1796 and 1799 test systems. Table F-3A and Figure F-5A list the connections available with 1799 test systems. The differences are listed in the remarks column.
4. The optional keyword STROBE in a SET PGEN statement has differing meanings in 1796 and 1799 test systems. An external PGEN start is available in 1799 systems (only). In 1796 test systems STROBE synchronizes SET PGEN execution to the next Driver Strobe signal.

1795 TEST SYSTEMS

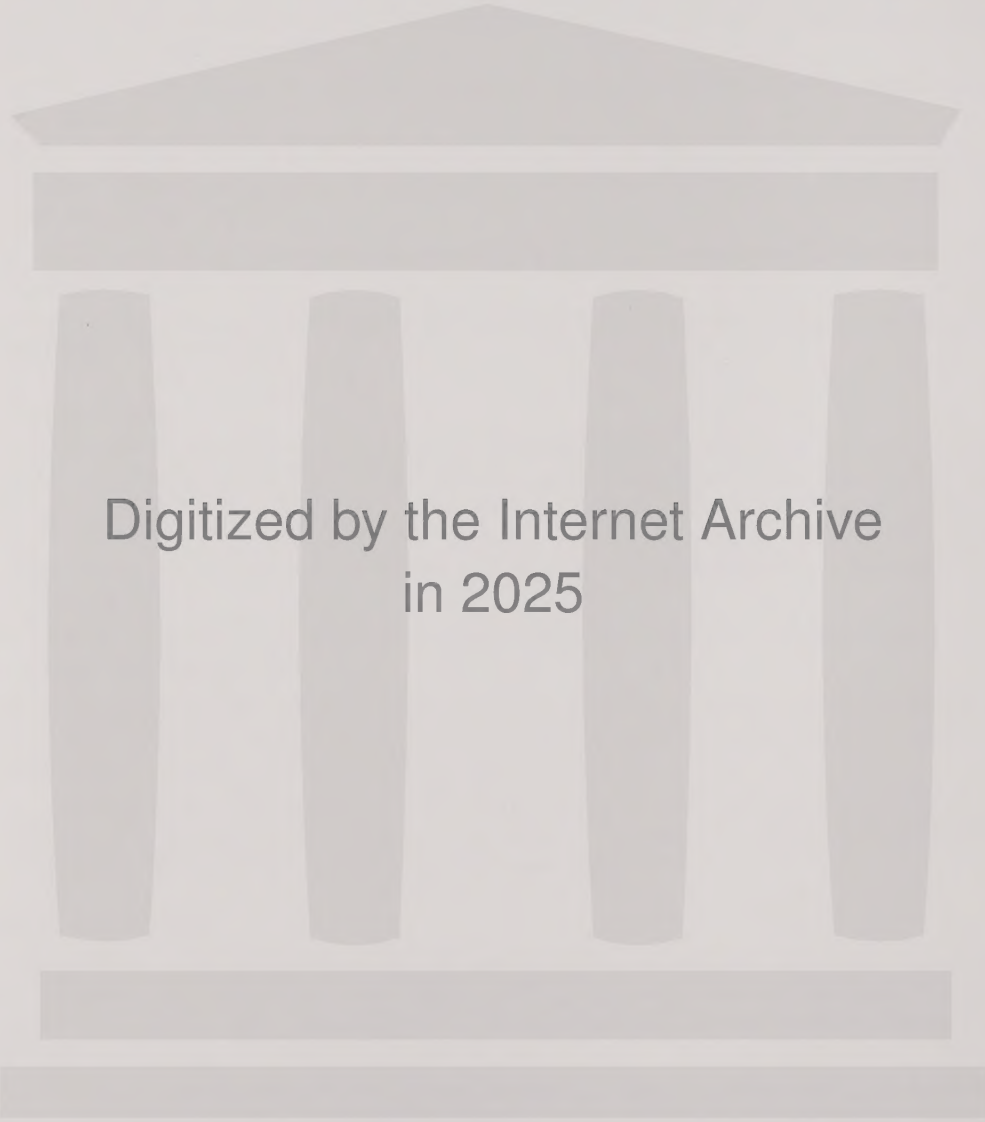
The 1795 is a low-cost digital test system without any built-in analog testing capability. Hybrid shelf scanner pins and device power supply scanning are **not** supplied. Test language statements for digital testing are described in Appendix D. The language is generally compatible with digital test statements used to prepare programs for 1799 test systems with the following exceptions:

1. The 1795-A version uses fixed (internally adjustable) driver/sensors with a single Logic Level Set and adjustable High levels only. Fully programmable driver/sensors are available with 1795-B versions only.
2. When using the fully programmable driver/sensors, each group of 12 pins **must** be assigned to the **same** Logic Level Set.
3. Universal Device Power Supplies are **not** available with 1795 test systems. Combinations of fixed and hardware adjustable Triple Power Supplies may be furnished (refer to Appendix D).

ALL TEST SYSTEMS

The following changes apply to all test systems:

1. The **BRANCH** command can be used to link multiple-CAPS sections together. Appendix C describes this and other alternatives of handling 2 or more program sections.
2. Appendix B describes the methods for involving **PAL8** routines as overlays or subroutines. With release VIII.3 software all test systems make use of the same symbols file called **OVERLY**.



Digitized by the Internet Archive
in 2025

https://archive.org/details/manualsplus_06652

SECTION 1

INTRODUCTION

- 1.1 Introduction**
- 1.2 The GR Test Systems**
- 1.3 System Manuals**
- 1.4 The Test Language**
- 1.5 System Description**

1.1 INTRODUCTION

This manual contains combined test programming instructions for GR Digital and Analog Test Systems. Programming differences unique to a specific type of system are given in the appendices. Appendix D covers methods of programming 1795/1799 test systems. Appendix F covers 1796 test systems with parts applicable to the 1799. The programming instructions are comprised of a description of the source language used for writing test programs and reference data. Using this data, test development personnel can develop source programs in a simple and straightforward manner to perform complex testing functions. The manual is organized as follows:

Section 1 contains introductory material and a brief description of the test language command structure.

Section 2 contains reference material that is used for program development.

Section 3 describes the various types of commands in detail. A good understanding of the information contained in this section is required before a programmer can write programs in the test language.

Section 4 briefly discusses the test considerations that a programmer must be familiar with before program development is started.

Section 5 contains complete sample programs which illustrate the technique of program construction.

Once the test program has been written, it is normally entered via the data terminal; however, the Teletype can be used for program entry. Details of program entry, editing, and test-system operation are contained in the Operating Instructions Manual for each Test System.

1.2 THE GR TEST SYSTEMS

GR Test Systems are the result of years of developmental experience with automatic systems for testing logic and analog circuit boards. Input test signals and device operating power to a unit under test (UUT)* are under the control of a test program which is written in an easy-to-use language. The test program also controls the selection of measurement and output devices which can be connected to UUT signals.

Most systems are operated from a single operating position, using a data-terminal display and control panel. Other input and output peripherals are either standard or optionally available depending on the system type. Interface to the UUT is made through a shelf adjacent to the operating position. A UUT is connected to the shelf through a device adaptor.

*A UUT is also called a Device Under Test (DUT)

1.3 SYSTEM MANUALS

The *GR Logic Circuit Analyzer Accessories and Serial List* contain a list of hardware, software, and instruction manuals shipped with each system. Included in the equipment shipment are all manuals for system items not manufactured by GenRad. System manuals, which provide detailed coverage of equipment manufactured by GenRad are listed and described in the *Operating Instructions Manual for the GR Test System*.

1.4 THE TEST LANGUAGE

The user directs test system operation through the use of test-language commands which are easy to learn and easy to use. A test program consists of a series of commands which are executed by the test system in a specific sequence to achieve a test objective. Three general types of commands are used: test equipment, arithmetic, and utility.

The test equipment commands control the power supplies, signal sources, and measurement equipment. The arithmetic commands provide for manipulation of data obtained from the execution of the test-equipment commands. Utility commands control program operations, such as printing error messages, introducing delays during test operation, terminating test programs, etc.

A program that is written in the test language is called a source program. Source programs are written and entered into the system using the EDIT Mode. When the source program is complete, the TRANS (translator) program generates a binary-coded version of the source program that can be executed by the system. The binary-coded program is then executed in the DIAGNOSIS Mode of the TEST or CAPS Monitor. Detailed descriptions of EDIT, TRANS, and DIAG Modes are included in the Operating Instructions.

Refer to the remaining sections in this manual for the information required for writing source programs. The remainder of this section contains general specifications of the equipment comprising the test system.

1.5 SYSTEM DESCRIPTION

The following paragraphs contain brief descriptions of the digital and analog test capabilities. Not all of the features listed below are available with each system type. Refer to the appropriate operating instruction and service manual for more detailed descriptions of the capabilities and the availability of peripheral equipment for each system.

Digital Test Unit (LGC)

The digital test unit includes the logic driver/sensor box and shelf assembly. Interface with the DUT is through the shelf input (driver) and output (sensor) pins. The maximum number of pins, and type (programmable or non-programmable) allowed depends on the system and system configuration. Each of the pins may be programmed as either an input or an output pin, reducing the number of interface adaptors required when testing different devices of the same family of units under test. In addition, any pin can be programmed to use one of two logic level sets, permitting circuits with mixed logic to be tested.

Voltage Level Programming Definitions:

VIH = Driver voltage when programmed high

VIL = Driver voltage when programmed low

VOH = Sensor threshold voltage; signals more positive than this level qualify as having valid high levels.

VOL = Sensor threshold voltage; signals more negative than this level qualify as having valid low signal levels.

The four voltage levels defined above comprise a "logic level set." Each driver/sensor pin of the test system may be assigned to one of two logic level sets allowed in any test program.

Table 1-1 lists the various test systems and indicates which logic-level set configuration may be provided.

Table 1-1

DRIVER/SENSOR CONFIGURATIONS

TEST SYSTEM	LOGIC - LEVEL SET	
	SINGLE	DUAL
1792A/B		X
1792C/D		X
1795A	X	
1795B	X	X
1796		X
1799	X	X

In all but 1795/1799 test systems any pin can be programmed to use either of the dual-logic-level sets (if supplied).

Each group of 12 pins can be assigned to either logic-level set in 1795/1799 test systems equipped with programmable driver/sensors. Designating **both** logic-level sets within a 12-pin group is **not** permitted. 1795/1799 systems with non-programmable driver/sensor pins have single logic-threshold.

Logic devices are tested by executing a series of logic test command statements. Each logic test command allows altering the state of any number of drivers and/or sensors. When the logic test command is executed, all assigned drivers assume their programmed states either simultaneously or sequentially in groups of 12, as controlled by the driver strobes. Table 1-2 indicates for each system the way the drivers are turned on. Similarly, all assigned sensors monitor the programmed voltage levels at the device outputs either simultaneously or sequentially in groups of 12, as controlled by the driver strobes. Table 1-2 indicates the way the drivers are turned on for each system type.

Table 1-2
DRIVER STROBED

System	Driver Threshold Type	Driver/Sensor Strobing
1792 C/D	Programmable	Simultaneously
1795-A	Non-programmable	Groups of 12
1795-B	Programmable	Simultaneously
1795-B	Non-programmable	Groups of 12
1796	Programmable	Simultaneously
1799	Programmable	Simultaneously
1799	Non-programmable	Groups of 12

Similarly, all assigned sensors monitor the programmed voltage levels at the device outputs either simultaneously or sequentially in groups of 12 as controlled by the sensor strobe(s) also shown in Table 1-2.

The time between the single or last sequential driver strobe and the first following sensor strobe is called the inter-strobe time (IST) and is programmable. Time between execution of the logic test commands is also programmable and is called the inter-test time (ITT). When desired, "delay" statements may be inserted between the logic test commands to effectively program inactive time between logic tests.

Function Generator Unit (ACS)

The function generator unit provides the analog test AC source (ACS) stimulus signal. Its main function in the test system is to generate sine, square, or triangular signal waveshapes over a wide voltage and frequency range. In addition, the ACS can add a DC voltage offset to the AC signal component, or provide a DC voltage level (without an AC component) test signal. Operational modes are programmable, as are the waveshape (function), signal amplitude, frequency, and DC offset voltage. Modes include: continuous, triggered, and gated.

Programmable Pulse Generator (PGEN)

The PGEN produces bursts of pulses for DUT stimulus. If desired, the pulses may be synchronized with the operation of driver/sensor circuits. Test program statements enable selection of: number of pulse transitions, delay between start of test and first pulse transition, pulse duration time, pulse separation time, pulse duration voltage, pulse separation voltage, and slope ($V/\mu\text{sec.}$).

Frequency Measurement Unit (FMU)

The frequency measurement unit is used for measuring frequency, frequency ratio, event counting, and time interval. Two inputs (A and B) are available and may be programmed for frequency and time measurements over a wide range.

DC to Digital Converter (DCV, DCI) [1792 D only]

The DC to digital converter provides a basic digital multimeter function for DC voltage and current measurements. One of four voltage or six current ranges is automatically selected when measuring DC values. Measurement data is rapidly obtained and returned to the test program for evaluation and/or display.

AC-DC Converter (ACV)

The AC-DC converter extends the basic DC to digital converter functions to include AC voltage measurements. It is calibrated over five automatically selected ranges to provide RMS measurements of sinewave input voltages.

Universal Power Supplies (UPS)

Universal power supply boxes each contain up to three programmable power supplies. Three slots are provided for power supply insertion, each having electrical connections which are compatible with all power supply modules. Four power supply modules are available, providing programmable voltages over 10 V, 20 V, 40 V, or 80 V ranges. Additional UPS boxes may be added as options depending on the system type to provide additional programmable power supply capacity.

The 80 V programmable power supply is larger than the other modules, requiring two slots, and has features which are not available in the other modules. Features of the 80 V unit include a current forcing, voltage limit/test mode, higher current resolution, and it can be floated at ± 250 V with respect to ground.

All programmable power supplies are controlled by a single logic card, regardless of the number of UPS boxes or programmable power supplies used.

Resistance Measurement Unit (RMU) [1792 D only]

The RMU provides a four-terminal test of resistance. In addition, the actual resistance value may be measured by a series of repeated tests under the control of the test program. A resistance test is rapidly executed by comparing DUT resistance with program-statement limit parameters.

Relay Drivers (RLY)

Relay drivers are standard in most systems and may be programmed ON or OFF. Test program applications include driving up to 24 relay coil or indicator lamp loads. The principal application is for UUT control.

Time Measurement Unit (TMU)

The TMU, with start and stop shelf inputs, is used in the time domain to test or measure the width, period, rise time, or fall time of pulses.

IEC (ASCII) Bus

The IEC (ASCII) interface allows external devices compatible with the IEEE-488 interface standard to be controlled by the system.

GR Pulse-Catching Probe

The GR Pulse-Catching Probe is primarily a diagnostic tool used in conjunction with GR CAPS software. However, it is possible to use the probe for making dc voltage measurements in the range of $\pm 30\text{V}$ nominal.

Triple Power Supply (TPS) (Except 1792 Systems)

The TPS consists of three resistor-programmable power-supply sections; one 0-7 V (5A), and two (0-20 V) (1.5A). Each section is independent and floating, and can supply either positive or negative voltages.

Precision Multimeter (DCV,DCI,RMU,ACV,DCVIF – 1796/1799 Only)

The multimeter option has the capability of digitizing voltage levels (of either polarity) to a resolution of 16 bits. The basic unit has DCV and DCI capability. Ac voltage, resistance measurement and high-compliance source (zener test) options can be added to the basic capability. Measurements can be conditioned with a 50-60 Hz integrator, or low-pass filter under software control by the basic capability. Also included is a peak detector (for either polarity).

The resistance measurement option (RMU) to the DCV/DCI unit provides for resistance measurement either by forcing current and measuring voltage, or the converse.

The ACV option to the DCV/DCI unit is an ac-to-dc converter. The dc output is applied to the DCV/DCI unit. A programmable 50Ω input terminator is available for higher frequency signals.

The DCVIF capability is a high-compliance current source option to the DCV/DCI unit. The DCVIF also requires the RMU option. Current can be sourced using a programmable current source to a compliance of 160 V. Test of zener diodes is a primary application for this option.

SECTION 2

REFERENCE MATERIAL

- 2.1 General
- 2.2 Definitions
- 2.3 Limits
- 2.4 Numerical Values
- 2.5 Comments
- 2.6 Character Set
- 2.7 Reserved Words
- 2.8 Branch Addresses And Labels
- 2.9 Registers

2.1 GENERAL

This section contains reference material that is frequently used when writing test programs. By correctly using the information, programming time and errors are kept to a minimum.

2.2 DEFINITIONS

In order to understand the test programming instructions contained throughout this manual, the reader should first become familiar with some of the more common terms used by programmers. Although the terms may seem to be quite basic to the experienced programmer, the test system is designed so that it can be used by inexperienced personnel; hence, the definitions provide the basis for understanding and using the test language. Table 2-1 contains the programming definitions for commonly used terms.

Table 2-1
PROGRAMMING DEFINITIONS

Term	Definition
Command	A statement comprising part of a source program that initiates some action. There are three types of source language commands: arithmetic, test equipment, and utility.
Editor	A program used to enter and/or make any necessary corrections to source program. (Refer to the Operating Instructions for a detailed description of the Editor.)
Error flag	A signal which produces a visual indication at the end of program execution, informing the operator that an error has been detected.
Execution order	The order in which statement functions are actually performed.
Execution time	The time required for the test equipment to complete a specified operation.
Floating point representation	A method of representing a numerical value which is either too large to be represented conveniently as an integer, or which is fractional or contains a fractional part.
Translation order	The physical ordering of statements in the source program.
Source language	A set of words, numbers and special characters used with a syntax designed for ease of use.
Source program	A program written in source language.
Translator	Converts source programs into a more compact form. The translator is also capable of detecting certain types of errors in the source program.

2.3 LIMITS

Limits in the COMP, TEST, and ATEST commands must be enclosed in inequality signs $<$ $>$. Limits can be expressed in several forms, but relate to an absolute value such as resistance, period or voltage, as follows:

1. **Directly:** <HI=5030 LO=4980>
2. **Variations from nominal:** <LIM=5000+30-20> (Limits are 5030 and 4980)
3. **Percentage variations from nominal:** <LIM=5000+6%-4%> (Limits are 5300 and 4980)

It is possible to specify only one limit. In this way, a single-side limit test is made: <HI=.006> or <LIM=10 k-2.7%> The numerical value stored in a register (see Table 2-6) can be specified as a HI or LO limit by using the register name as the limit: <HI=REG-3 LO=REG-4> or <HI=DCV>

2.4 NUMERICAL VALUES

Unless specified otherwise, parameters will have the following units assumed:

Time:	Seconds
Voltage:	Volts
Resistance:	Ohms
Current:	Amperes
Frequency:	Hertz

The following suffixes can be used when numerical values are written. They must follow the numerical digits immediately, i.e., 10MEG *not* 10 MEG *Insertion of a space will result in an error.*

MEG	X10 ⁶
K	X10 ³
M	X10 ⁻³
U	X10 ⁻⁶
N	X10 ⁻⁹
P	X10 ⁻¹²
E	(see note)

NOTES

1. Numerical values can also be specified in exponential form; i.e., 2.3E5,, -1.7E-6, etc. The letter E is followed directly by the decimal exponent, except for a possible minus sign. Insertion of a space immediately following the E will cause a translator error.
2. Registers may be used in place of floating point values except in DELAY command statements and DLY, IST, and ITT specifications.
3. Floating point values which are outside the programmable range of test equipment are not detected by the translator (TRANS) during the program translation. This type of error is detected at program execution time (DIAGNOSIS Mode) when attempting to execute the instruction. An error of this type will cause an appropriate system error message to be displayed (see Operating Instructions for a description of TRANS and DIAG and a listing of system error messages).

2.5 COMMENTS

Comments appear only in source listings as documentation aids and do not affect the binary program. Programmer comments can be inserted anywhere in the test program between slash marks (/). All characters following a slash will be ignored until a second slash is found irrespective of carriage-return characters. It should be noted that if the second slash is inadvertently omitted, the rest of the program will be treated as a comment.

2.6 CHARACTER SET

The test language character set contains all letters and numerals and the following characters: [] ! " # \$ % ' () * + , - . / : ; < > = ? . Table 2-2 explains the function of the special characters.

Table 2-2
SPECIAL CHARACTERS

Character	Explanation of Use
//	Delimit comments
[]	Enclose branch destination command label
;	Delimit end of command
:	Delimit label (command number) or separator between high and low sides in scanner pin specifications.
,	Separates numbers in list
-	Minus sign, or causes an inclusion of pin numbers for logic, relay, or scanner pin lists
()	Enclose lists
#	Has a meaning of "all but" in logic or relay list
< >	Enclose limit specification
*	Indicates "no change" in a terminal specification
' '	Enclose character string
%	Prefix 3-digit octal representation of ASCII character or function in character string
\$	Abbreviation for SET LGC command

2.7 RESERVED WORDS

The reserved word set separates into four categories: test equipment (Table 2-3), functions (Table 2-4), parameters (Table 2-5), and registers (Table 2-6). Where applicable, correlation to the command type and test equipment is provided, allowing reference to Section 3, Description Of Commands. Reserved words can be abbreviated to the underscored characters.

Table 2-3
TEST EQUIPMENT RESERVED WORDS

Word	Description
<u>ACS</u>	AC source (ACS)
<u>ACV</u>	AC voltmeter (AC-DC converter)
<u>BUS</u>	IEC (ASCII) bus
<u>CAPS</u>	CAPS simulator and diagnostics
<u>COUNT</u>	Count of pulses (FMU)
<u>DCI</u>	DC current meter (DC-digital converter)
<u>DCV</u>	DC voltmeter (DC-digital converter)
<u>DCVIF</u>	High compliance current source, (precision DC-digital converter)
<u>DCVPRB</u>	Dc voltage (Pulse-catching probe)
<u>FAST</u>	<u>Versatile Driver/Sensor Assembly</u> (high-speed testing)
<u>FMU</u>	Frequency measurement unit
<u>FRATIO</u>	Frequency ratio (FMU)
<u>FREQ</u>	Frequency (FMU)
<u>LGC</u>	Logic drivers and sensors
<u>MASCN</u>	Analog Scanner (Measurement channel A)
<u>MBSCN</u>	Analog Scanner (Measurement channel B)
<u>PASCN</u>	Power Supply Scanner (Channel A)
<u>PBSCN</u>	Power Supply Scanner (Channel B)
<u>PCSCN</u>	Power Supply Scanner (Channel C)
<u>PDSCN</u>	Power Supply Scanner (Channel D)
<u>PESCN</u>	Power Supply Scanner (Channel E)
<u>PGEN</u>	Pulse generator
<u>POWER</u>	All power supplies
<u>PS-i</u>	Power supply in slot 'i'

Table 2-3 (cont)
TEST EQUIPMENT RESERVED WORDS

Word	Description
<u>PSI</u>	Power supply ammeter
<u>PSV</u>	Power supply voltmeter
<u>RECORD</u>	Reference to most recently begun CAPS area
<u>RLY</u>	Relay
<u>RMU</u>	Resistance or resistance measurement unit (RMU)
<u>SASCN</u>	Analog Scanner (Source channel A)
<u>SBSCN</u>	Analog Scanner (Source channel B)
<u>SYNC</u>	Sync pulse
<u>TIME</u>	Time (FMU)
<u>TMU</u>	Time measurement unit
<u>XMEAS</u>	External Measurement Device (-1, -2, or -3)
<u>XSOURCE</u>	External Source Device

Table 2-4
FUNCTIONS RESERVED WORDS

Word	Command Type	Description
<u>ABSOLUTE</u>	Arithmetic	Replace signed contents of AR with absolute value
<u>ADD</u>	Arithmetic	Add to contents of AR
<u>ASSIGN</u>	Test equip.	Assign initial conditions
<u>ATAN</u>	Arithmetic	Replace contents of AR with its arc tangent
<u>ATEST</u>	Arithmetic	Arithmetic test (comparison and possible error flag)
<u>BEGIN</u>	Test equip.	Start data collection for CAPS simulator; start FAST block
<u>BLOCK</u>	Utility	Start a block
<u>BRANCH</u>	Utility	Program transfer to a specified command number
<u>CALL</u>	Utility	Subroutine call
<u>CCOMP</u>	Utility	Character string compare
<u>CHECK</u>	Utility	Check that specified condition is met and take appropriate action according to result
<u>CLEAR</u>	Test Equip.	Disconnect, take off line, set to zero
<u>CLEAR ALL</u>	Test equip.	Disconnect all scanner contacts that are set and clear all devices
<u>CLOSE</u>	Utility	Enters logging file into directory
<u>COMP</u>	Arithmetic	Comparison without possible error flag
<u>COS</u>	Arithmetic	Replace contents of AR with its cosine
<u>DELAY</u>	Utility	Programmable delay
<u>DIV</u>	Arithmetic	Divide (refers to contents of AR)
<u>DO</u>	Utility	Start a DO loop
<u>DONE</u>	Utility	End a DO loop
<u>END</u>	Utility	Test program terminator
<u>ERRET</u>	Utility	Error-Return from subroutine
<u>ESCAPE</u>	Utility	Terminate and branch out of DO loop
<u>EXEC</u>	Utility	Execute special PAL-coded routine
<u>EXP</u>	Arithmetic	Replace contents of AR with Log base e raised to the power of AR
<u>FINISH</u>	Test equip.	End data collection for CAPS simulator; end FAST block
<u>FORMAT</u>	Utility	Specify output format
<u>IGNORE</u>	Test equip.	Ignore pins in logic test programs
<u>LD (LOAD)</u>	Arithmetic	Load (into the AR register)
<u>LOG</u>	Arithmetic	Replace contents of AR with its natural Log
<u>MCOMP</u>	Test equip.	Measure and compare (same as TEST except no error flag is generated if test conditions are not met)
<u>MEAS</u>	Test equip.	Measure (and store) a specified parameter
<u>MPY (MULT)</u>	Arithmetic	Multiply (contents of AR and a second operand; result placed in AR)

Table 2-4 (Cont.)
FUNCTIONS RESERVED WORDS

Word	Command Type	Description
<u>NOP</u>	Utility	No Operation
<u>OPEN</u>	Utility	Create logging file
<u>PAUSE</u>	Test equip.	Pause until CONTINUE button is pressed
<u>PNAME</u>	Utility	Assign symbolic name to pin number
<u>PULSE</u>	Test equip.	Generate logic pulses
<u>READ</u>	Utility	Read from device into register
<u>REASSIGN</u>	Test equip.	Modify initial Driver/Sensor functions
<u>RESERVE</u>	Utility	Reserve core storage
<u>RETURN</u>	Utility	Return from subroutine
<u>SCOMP</u>	Utility	Compare two character strings, displaying error message or taking optional error branch if not identical
<u>SET</u>	Test equip.	Set-up test or measurement initial conditions; set FAST options; transmit IEC Bus commands
<u>SIN</u>	Arithmetic	Replace contents of AR with its sine
<u>SMOVE</u>	Utility	Move specified string of characters to specified location
<u>SQUARE</u>	Arithmetic	Square contents of AR
<u>SQRT</u>	Arithmetic	Replace contents of AR with the square root of the contents of AR
<u>STEST</u>	Utility	Compare two character strings, displaying error message or taking optional error branch if not identical (also increments FAIL Register)
<u>STORE</u>	Arithmetic	Store the contents of AR in the designated register
<u>SUBTRACT</u>	Arithmetic	Subtract from the contents of AR
<u>TEST</u>	Test equip.	Execute test and display result (PASS or FAIL)
<u>WRITE</u>	Utility	A named device writes the items named in a list

Table 2-5
PARAMETERS RESERVED WORDS

Word	Description
<u>A</u>	Select analog probe relay A
<u>ADIR</u>	Bypass channel A input attenuator (FMU)
<u>AT</u>	Scanner terminal specification follows
<u>B</u>	Select analog probe relay B
<u>BDIR</u>	Bypass channel B input attenuator (FMU)
<u>C</u>	Close (RLY), or read character string; select analog probe relay C
<u>CLK</u>	Select clock input (FMU)
<u>CMSR</u>	Select analog probe capacitively-coupled measurement mode
<u>CONT</u>	Continuous mode (PGEN)
<u>COUNT</u>	Count of pulses (FMU)
<u>D</u>	Select analog probe relay D
<u>DC</u>	DC output (no waveform) ACS)
<u>DCL</u>	IEC Bus universal command: <u>Device C</u> lear
<u>DELAY (DLY)</u>	Delay (PS, DCV, DCI, ACV)
<u>DEV</u>	Device specification follows
<u>DIRMSR</u>	Select analog probe direct-coupled measurement mode
<u>DIRSRC</u>	Select analog probe direct-coupled measurement mode
<u>EXIT</u>	Terminates high-speed DO loop on specified sensor state
<u>EXTS</u>	External start (DCV/DCI/DCVIF/RMU)
<u>F</u>	Frequency (ACS)
<u>FILE</u>	Logging file
<u>FIRST</u>	First interrupting device
<u>FORCE</u>	Force (ACV, DCV, DCI)
<u>GATE</u>	Gated mode (ACS)

**Table 2-5 (Cont.)
PARAMETERS RESERVED WORDS**

Word	Description
<u>GET</u>	IEC Bus addressed command: <u>G</u> roup <u>E</u> xecute <u>T</u> rigger
<u>GSENSE</u>	Connection for Guard Sense terminal (RMU)
<u>GTL</u>	IEC Bus addressed command: <u>G</u> o <u>T</u> o <u>L</u> ocal
<u>GUARD</u>	Connection for Guard terminal (RMU)
<u>HI</u>	High limit follows
<u>I</u>	Current specification follows (PS)
<u>IA</u>	List of drivers for logic set A (LGC)
<u>IB</u>	List of drivers for logic set B (LGC)
<u>IH</u>	Input high specification follows (LGC)
<u>IL</u>	Current-locked mode (PS) or input low specification
<u>IM</u>	Negative current terminal connection (RMU)
<u>INT</u>	Causes a time interval measurement to use Channel A
<u>INTG</u>	Integrate mode (DCV/DCI)
<u>IP</u>	Positive current terminal connection (RMU)
<u>IST</u>	Inter-strobe time specification follows (LGC)
<u>ITT</u>	Inter-test time specification follows (LGC)
<u>K</u>	10 ³
<u>LIM</u>	Limit
<u>LLO</u>	IEC BUS universal command: <u>L</u> ocal <u>L</u> ock <u>O</u> ut
<u>LMTCHK</u>	Limit check
<u>LO</u>	Low limit follows
<u>LPF</u>	Low-pass filter mode (DCV/DCI)
<u>LOWZ</u>	Low impedance (ACV)
<u>LPT</u>	Line printer
<u>M</u>	10 ⁻³
<u>MAX</u>	Maximum expected voltage measurement specification (DCV/DCI)
<u>MEG</u>	10 ⁶
<u>MLA</u>	IEC Bus universal command: <u>M</u> y <u>L</u> isten <u>A</u> ddress
<u>MSA</u>	IEC Bus universal command: <u>M</u> y <u>S</u> econdary <u>A</u> ddress
<u>MTA</u>	IEC Bus universal command: <u>M</u> y <u>T</u> alk <u>A</u> ddress
<u>N</u>	10 ⁻⁹
<u>N</u>	Transition count (PGEN, Logic)
<u>NEGATIVE</u>	Reverse direction/polarity of forced current/voltage (RMU)
<u>NPEAK</u>	Negative peak detection mode (DCV/DCI)
<u>NO</u>	No output (ACS)
<u>O</u>	Open (RLY)
<u>OA</u>	List of sensors for logic set A (LGC)
<u>OB</u>	List of sensors for logic set B (LGC)
<u>OCTAL</u>	Prefixes data specification in octal format
<u>OFF</u>	All power supplies disabled (PS)
<u>OH</u>	Output high specification follows (LGC)
<u>OL</u>	Output low specification follows (LGC)
<u>ON</u>	All power supplies enabled (PS)
<u>P</u>	10 ⁻¹²
<u>PPEAK</u>	Positive peak detection mode (DCV/DCI)
<u>PREC</u>	Precision with which a FMU measurement is to be made
<u>PRESET</u>	Use previous parameter setting for next measurement (DCV/DCI/DCVIF/RMU)
<u>PRINT</u>	Pre-assigned listing device
<u>PROBE</u>	Analog probe(s)
<u>PTP</u>	High-speed paper-tape punch
<u>PTR</u>	High-speed paper-tape reader
<u>R</u>	Read numeric data, or REference terminal

Table 2-5 (Cont.)

PARAMETERS RESERVED WORDS

Word	Description
<u>RCSRC</u>	Select analog probe RC-coupled source mode
<u>RECT</u>	Rectangular waveform (ACS)
<u>RELPRO</u>	Make scanner connections in Relay Protect mode
<u>RNG</u>	Range specification follows (DCV, DCI, ACV)
<u>S</u>	Slope (PGEN)
<u>SBIT</u>	Specify single-bit in character string
<u>SCAN</u>	Scanner system
<u>SDC</u>	IEC Bus addressed command: <u>S</u> elect <u>D</u> evice <u>C</u> lear
<u>SHELF</u>	Connect source output to unscanned pins (PGEN/ACS)
<u>SLOW</u>	Slow time constant (ACV)
<u>SPD</u>	IEC Bus universal command: <u>S</u> erial <u>P</u> oll <u>D</u> isable
<u>SPE</u>	IEC Bus universal command: <u>S</u> erial <u>P</u> oll <u>E</u> nable
<u>STROBE</u>	PGEN will be started by next D/S strobe (PGEN)
<u>T-i</u>	Scanner system multiplexer terminals
<u>TEC</u>	Data terminal
<u>TD</u>	Pulse time duration (PGEN)
<u>TH</u>	Transitional high specification follows (CAPS)
<u>TIN</u>	Terminate input (FMU)
<u>TINA</u>	Terminate input, channel A only (FMU)
<u>TINB</u>	Terminate input, channel B only (FMU)
<u>TL</u>	Transitional low specification follows (CAPS)
<u>TR</u>	Places (FMU) in tracking mode
<u>TRI</u>	Triangular waveform (ACS)
<u>TRIG</u>	Triggered mode (ACS)
<u>TRIGA</u>	Trigger Channel A (FMU)
<u>TRIGB</u>	Trigger Channel B (FMU)
<u>TS</u>	Pulse separation (PGEN)
<u>TTY</u>	Teletype
<u>TX</u>	Transitional unknown specification follows (CAPS)
<u>T0</u>	Delay before first pulse (PGEN)
<u>U</u>	10^{-6}
<u>USER</u>	Signals from high-speed controller to UUT
<u>V</u>	Voltage specification follows (PS)
<u>VCG</u>	Voltage controlled frequency generation (ACS)
<u>VD</u>	Voltage duration (PGEN)
<u>VF</u>	First voltage specification follows (TMU)
<u>VH</u>	Verify output high specification follows (CAPS)
<u>VIHA</u>	Driver high voltage level for logic set A (LGC)
<u>VIHB</u>	Driver high voltage level for logic set B (LGC)
<u>VILA</u>	Driver low voltage level for logic set A (LGC)
<u>VILB</u>	Driver low voltage level for logic set B (LGC)
<u>VL</u>	Voltage-locked mode (PS), or verify output low specification follows (CAPS)
<u>VM</u>	Negative potential terminal connection (RMU)
<u>VOHA</u>	Sensor high voltage level for logic set A (LGC)
<u>VOHB</u>	Sensor high voltage level for logic set B (LGC)
<u>VOLA</u>	Sensor low voltage level for logic set A (LGC)
<u>VOLB</u>	Sensor low voltage level for logic set B (LGC)
<u>VP</u>	Positive potential terminal connection (RMU)
<u>VS</u>	Voltage separation (PGEN), or Second voltage specification follows (TMU)
<u>Z</u>	Issue Sensor Strobe immediately after Driver Strobe (high-speed controller)
<u>ZERO</u>	Remove voltage offset before making measurement (DCV)

Table 2-6

REGISTERS RESERVED WORDS

Word	Description	Unit
<u>AR</u>	The arithmetic register	—
<u>ACV</u>	AC voltage	Volts rms
<u>CHAR</u>	Character string (up to 20 characters)	—
<u>CONTINUE</u>	CONTINUE button counter	—
<u>COUNT</u>	Event count	—
<u>DCI</u>	DC current	Amperes
<u>DCV</u>	DC Voltage	Volts
<u>DCVIF</u>	DC Voltage, high-compliance current source measurement	Volts
<u>DCVPRB</u>	Pulse-Catching Probe DC Voltage	Volts
<u>FAIL</u>	Test fail counter	—
<u>FRATIO</u>	Frequency ratio	—
<u>FREQ</u>	Period	Seconds
<u>PSI</u>	Power supply DC current	Amperes
<u>PSV</u>	Power supply voltage	Volts
<u>REG-i</u>	Registers (1 — 100)	—
<u>RMU</u>	DC resistance	Ohms
<u>SREG</u>	String register (characters)	—
<u>TIME</u>	Time	Seconds
<u>TMU</u>	Time	Seconds
<u>USOPT1</u>	Toggle register for USER OPTION 1 switch	—
<u>USOPT2</u>	Toggle register for USER OPTION 2 switch	—

2.8 BRANCH ADDRESSES AND LABELS

Labels are optional numbers that precede test statements (commands). They allow reference to a command for program branching, calls, or test identification on an error message display. Labels (also called test numbers) are numbers ranging from 1 through 8,388,607, containing no blanks, and terminated with a colon (:). A label must precede each test statement referenced. When an error is detected in a test statement that does not include a label, the next previous label is displayed along with a number indicating the number of test statements beyond the label where the error is detected. The use of labels increases the size of the binary-coded program with a corresponding increase in program execution time.

Branch addresses may be included in SET, MEAS, ATEST, and TEST commands. If they are included, an error condition detected when executing the command will cause program control to transfer to the statement pointed to by the branch address. If no branch address is present, a standard error-handling procedure will be invoked. The branch address is contained within brackets. Note that the branch address must not be confused with the BRANCH utility command described in Section 3.

Branch addresses are also included in all COMP, CCOMP and MCOMP commands. If the test conditions are not satisfied, the program branches to the stated address; the branch address is a required parameter for these commands. These commands differ from the SET, ATEST and TEST command executions by not generating an error flag (FAIL indication on the operator's control panel).

Branch addresses in MEAS commands cause a branch if, for whatever reason, the measurement cannot be made. The error flag is not set if the branch is taken.

Branch addresses, included in CALL commands, can be passed as error returns for use by SET, MEAS, TEST, ATEST, COMP, CCOMP, and MCOMP commands in the CALLED routines. Commands in a CALLED routine specify the passed branch address thusly:

[ERRET] or more simply []

The CALLED routine may pass control directly to the error return via the command:

ERRET;

2.9 REGISTERS

2.9.1 Registers Available

Registers are memory locations that the test program uses during execution for temporary storage. The various registers and their functions are described below.

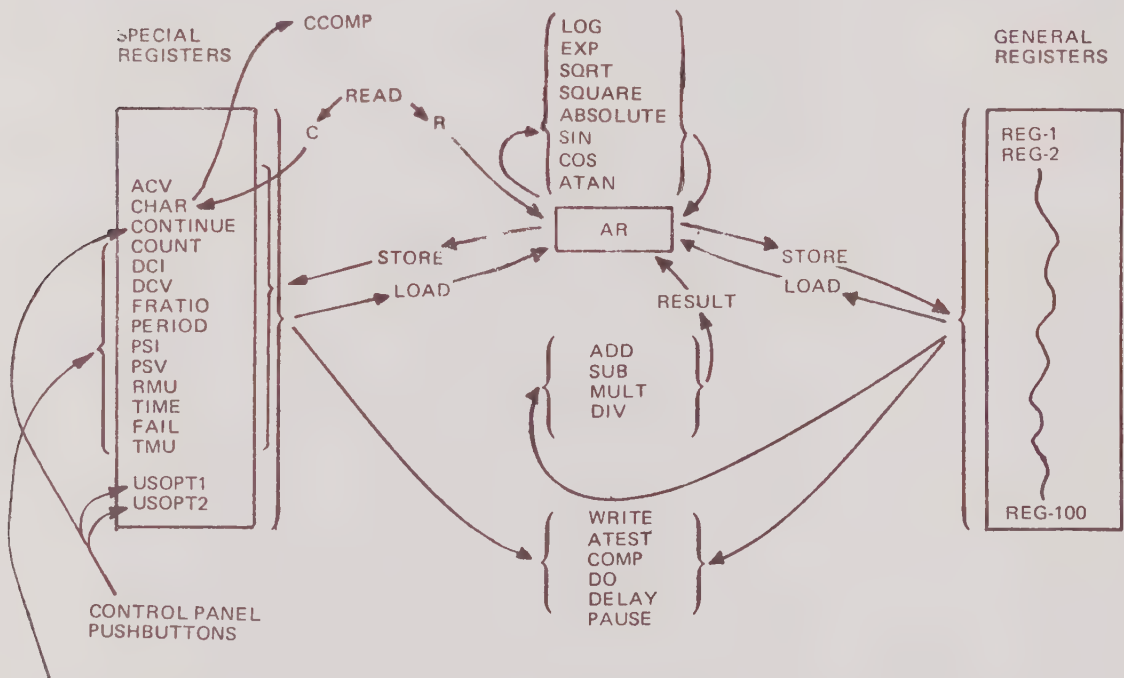
Table 2-7
REGISTERS

Name	Description
AR	Arithmetic Register — This is the working register for all arithmetic operations and operations involving measurements by the test system. The most recent arithmetic result or measured value appears in this register.
REG-1 through REG-100	General Registers — These registers are available for any test program function.
ACV	AC Voltage — Contains the floating-point result (in volts RMS) of the AC voltage measurement effected by the last TEST/MEAS ACV command executed.
CHAR	Character — Provides storage for a character string of up to 20 characters.
CONTINUE	Continue — Contains a binary count which is incremented each time the CONTINUE button is pressed during test program execution.
COUNT	Count — Provides storage of the event count effected by the last TEST/MEAS COUNT command executed.
DCI	DC Current — Contains the floating-point result (in DC amperes) of the DC current measurement effected by the last TEST/MEAS DCI command executed.
DCV	DC Voltage — Contains the floating-point result (in volts DC) of the DC voltage measurement effected by the last TEST/MEAS DCV command executed.
FAIL	Fail Register — Contains a binary count of individual test failures detected during test program execution. At the end of test program execution, when no operating under CAPS, the FAIL indicator illuminates if the FAIL register contents are non-zero, or the PASS indicator illuminates if the contents are zero.
FRATIO	Frequency Ratio — Contains the floating point result of the frequency ratio measurement effected by the last TEST/MEAS FRATIO command executed.
FREQ	Frequency — Contains the floating-point result (in Hz) of the frequency measurement effected by the last TEST/MEAS FREQ command executed.
PSI	Power Supply Current — Contains the floating point result (in DC amperes) of the power supply current measurement effected by the last TEST/MEAS PSI command executed.
PSV	Power Supply Voltage — Contains the floating-point result (in volts DC) of the power supply voltage measurement effected by the last TEST/MEAS PSV command executed.
RMU	Resistance Measurement Unit — Contains the floating-point result (in ohms) of the resistance measurement effected by the last MEAS RMU command executed.
TIME	Time — Contains the floating-point time (in seconds) result of the time measurement effected by the last TEST/MEAS TIME command executed.
TMU	Time Measurement Unit — Contains the floating-point time (in seconds) result of the time measurement effected by the last TEST/MEAS TMU command executed.
USOPT1 and USOPT2	User Option Registers — These two-state reference registers are toggled (complemented) with each operator actuation of the corresponding USER OPTION pushbutton, thereby, providing a means, through appropriate branching commands, for operator direction of the test program run. The initial state of these registers is selected through the monitor when DIAG is entered. USOPT1 and 2 can also be program controlled (see para. 3.4.16).

2.9.2 Register Operations

Various programmed operations can be performed using the contents of registers. Operations include the arithmetic and some of the utility commands described in para. 3.3 and para. 3.4. Figure 2-1 illustrates the program access to these registers using the commands. Note that these operations are in addition to the programmed TEST/MEAS operations that store data in these registers, and the test equipment commands that reference the stored data as parameters.

The program can STORE or LOAD the contents of any of the registers from AR. SQRT, SQUARE, or ABSOLUTE commands operate only on the contents of AR. ADD, SUB, MULT, and DIV use the AR and any other register as operands (AR is assumed but a second register must be specified); the operation result is placed in the AR. ATEST and COMP test or compare the contents of any register with command parameters. COMP compares the contents of only the CHAR register with command parameters (character strings). WRITE outputs the contents of any register to a named device. DO, and PAUSE use the numeric value contained in any register when named in the parameter field of the command. The READ command inputs C (character) data to the CHAR register, or R (real numeric) data to the AR.



NOTE: Data is also stored via the AR register to the indicated special registers by applicable MEAS and TEST commands (refer to Table 2-7).

Figure 2-1. Register Operations.

SECTION 3

DESCRIPTION OF COMMANDS

- 3.1 General
- 3.2 Test Equipment Commands
- 3.3 Arithmetic Commands
- 3.4 Utility Commands
- 3.5 MACRO Programming Aid

3.1 GENERAL.

The test language consists of three categories of commands: test equipment, arithmetic, and utility commands. Test equipment commands control the power supplies, signal sources and measurement equipment. Arithmetic commands provide for manipulation of data obtained from the execution of the test equipment commands. Utility commands provide control over program operation, such as printing error messages, introducing delays during test operation, terminating test programs, etc. A test program is simply a series of these commands which are executed in a specific sequence to achieve a desired test objective.

The format of the commands is included with the description of each category. Note that all commands must terminate with a semi-colon (;). A command statement may be prefixed with a numeric label, ranging from 1 to 8000000, terminated by a colon (:).

The following paragraphs describe the arithmetic, utility, and test-equipment commands. The format of each category is also described. The arithmetic commands include such operations as Load, Store, Square Root, Arithmetic Test, and others. The utility commands provide for error-message and TTY printouts, and such varied operations as pausing, ending, report initiation, logging messages, etc. The main test language category is test-equipment commands. This category includes, among other commands, commands for control of such stimulus devices as ac sources and programmable power supplies, and measurement units such as frequency- and resistance-measurement units, ac and dc voltmeters, etc.

The examples illustrated in this section are not intended to convey all the possible command configurations, only techniques. Several techniques are described that aid in both shortened execution time and compact program development. It is suggested that this entire section be read thoroughly before a specific program development is attempted.

3.2 TEST EQUIPMENT COMMANDS

3.2.1 General

Test equipment command statements consist of four fields which follow the statement label (if used):

1. **Command Function Field.** This field consists of one word such as SET, CLEAR, MEAS, TEST, ASSIGN, etc.
2. **Test Equipment Device Field.** This field identifies the device/measurement: PS-1, ACV, etc.
3. **Terminal Field.** This field contains a list of terminals (shelf pins) enclosed by parentheses and preceded by a function-identifying prefix. A device adaptor provides the required connections between the terminals and the DUT interface connector.
4. **Parameter Field.** This field contains any other necessary qualifications, depending on the command and test, but includes such parameters as limits, branch addresses, and optional parameters such as range and programmable delays, etc.

3.2.2 Test and Measurement Commands

Two types of commands are provided for DUT measurements or tests: MEAS and TEST. The MEAS command causes complete measurement of a numerical value for the desired parameter. The measured value(s) is then placed in one or more registers (refer to Table 2-6) for comparison and/or analysis. For ACV, DCV, and DCI measurements, an optional RNG (Range) specification parameter in a MEAS command will reduce execution time, provided that the range included is the correct one. If no range is specified, the test equipment will have to search by trial and error for the correct range. This is a relatively time-consuming procedure.

Note that in addition to the test equipment commands which are grouped by test equipment or function, commands such as SET, ASSIGN, TEST, etc. establish the test equipment configuration or set-up before the test and measurement functions are performed.

Established connection configuration and set-up of individual test equipment can be explicitly rescinded by a CLEAR command directed toward the particular test equipment. A general CLEAR ALL command invokes a complete sequence of explicit CLEAR commands.

3.2.3 Explanation of Command Examples

In the test statement examples that use the letters a, b, c, etc. for specifying parameters, the letters represent floating-point numbers; i.e., numbers which are fractional or have a fractional part, or numbers which are too large to be expressed conveniently as integers.

Letters i, j, k, l, m, n represent positive integers.

Where one item can be chosen from several, they are listed in a vertical column. When one of the legitimate possibilities is to include none of these items, the word 'blank' is included in the list.

3.2.4 Terminal Designations

System test equipment either exercises or monitors the signals at the terminals (shelf pins) of the test system. Shelf pins are specifically designated in the test equipment command statement by the terminal field which has the following form.

prefix (pin-list)

The prefix in the terminal field associates with the device specified in the test equipment device field of the command statement. The following prefixes, which are described in Table 2-5, apply when LGC (logic drivers and sensors) is the specified device.

IA	OA	IH	OH	VH	TH	TX
IB	OB	IL	OL	VL	TL	

In the 1792C test system, the additional prefix AT is used in the terminal field when the specified device is TMU (time measurement unit).

In the 1792D test system, the AT prefix is also used in terminal fields when specified analog test equipment devices are to utilize the source, measurement, or power supply pin-scanners. Also in the 1792D test system, the additional prefixes of O and C are used when the specified device is RLY (relay drivers).

The *pin-list*, when preceded by a *prefix other than AT*, is a list of shelf pin numbers (or mnemonic pin names previously established by the ASSIGN PNAME utility command) enclosed between parentheses. In multi-pin lists, individual pins are separated by commas (,) as shown in the following typical command statement.

TEST LGC IH(2,4,6) IL(8,10,12) OH(7) OL(9,11);

A hyphen (—) in a pin-list means the same as the word *thru* and is used to include the numerical sequence of pin numbers between those specified in the specification. In the following command statement, the hyphen in the pin list indicates 12 pins (1 through 12) are assigned as inputs in logic level set A.

```
ASSIGN LGC IA(1-12) VIHA=15 VILA=1 VOHA=12 VOLA=2;
```

A sharp symbol (#) in a pin-list means the same as the term *all but* and explicitly specifies the condition for all active pins except those listed (the conditions of the listed pins remain unchanged from previous specification). For example: in a program that contained the preceding ASSIGN LGC statement, which established 12 active input pins (1-12), the following SET LGC statement would explicitly set input pins 2 through 11 high and leave input pins 1 and 12 in their previous state.

```
SET LGC IH(#1,12);
```

A *pin-list* preceded by the **AT** prefix has the following format where a colon (:) separates a primary-pin-specification from secondary-pin-specification, all enclosed between parentheses.

```
AT (primary-pin-specification: secondary-pin-specification)
```

Each pin specification is a list of shelf pin numbers (or mnemonic pin names previously established by the ASSIGN PNAME utility command) and may contain the comma (,) and hyphen (—) conventions previously described. The following is typical of a command statement that uses the power supply scanner to connect the high-side terminal of the power supply to pins 1 and 12 (primary-pin-specification) and the low-side terminal of the power supply to pins 5 through 8 (secondary-pin-specification).

```
SET PS-3 AT(1,12:5-8) V=-15 I=.05;
```

An asterisk (*) for one of the pin specifications in the pin-list means that the pin-specification remains connected as previously commanded for the particular test equipment device and only the other connection is changed as specified. For example: the following series of commands effects two voltage measurements using the same shelf pin (pin 3) in both measurements as the high-side input to the DCV.

```
MEAS DCV AT (3:7);  
MEAS DCV AT (*:8);
```

If the command shown below were added to the above series of commands, the high-side input to the DCV would be changed to pin 4 and the common input would remain at pin 8 as previously specified.

```
MEAS DCV AT (4);
```

A numeric zero (0) can be listed for the pin specifications to cause the pin-scanner to physically disconnect a test equipment device from shelf pins, as illustrated in the following command.

```
SET XMEAS-1 AT(0:0);
```

The secondary-pin-specification is not required in the pin lists of command statements for test equipment devices that have only one scannable terminal. For example, execution of the following command measures the AC voltage between pin 8 and analog ground (analog ground is a permanent reference for the ACV device).

```
MEAS ACV AT (8);
```

3.2.5 Earmarking CAPS Stimuli [CAPS – Equipped Test Systems Only]

The following commands are used in systems equipped with CAPS software to delineate CAPS sections and the recording of test results-of-interest to CAPS generated diagnostics.

BEGIN CAPS 'section-name';

The BEGIN CAPS command specifies the beginning of a CAPS section of the test program. Up to six alphanumeric characters may comprise the section-name. This command also implies a CLEAR ALL command (except no registers are cleared) and a BEGIN RECORD command. An ASSIGN LGC statement must follow the BEGIN CAPS command and precede all other LGC commands of an opened CAPS recording area. The BEGIN CAPS command also zeroes the test step counter which is then incremented by each subsequent logic test in CAPS recording sections. The step counter is displayed as the CAPS # in status messages.

FINISH CAPS;

The FINISH CAPS command terminates a CAPS section and in the DIAGNOSIS Mode, closes the file previously opened by a BEGIN CAPS command. This command also implies a FINISH RECORD command and terminates program execution in the DIAGNOSIS Mode if CAPS failures were recorded and no non-CAPS failures occurred.

FINISH RECORD;

Within a CAPS section, the FINISH RECORD command suspends the recording of test results in the DIAGNOSIS Mode and the CAPS simulation in the PREPARE Mode to permit subsequent non-CAPS related commands in the test program.

BEGIN RECORD;

Within a CAPS section, the BEGIN RECORD command re-activates the CAPS simulator/diagnostic process for subsequent test program commands. During program execution in the DIAGNOSIS Mode, a failure detected within a CAPS RECORDing area (when the non-CAPS portion of the program is failure free) causes a link to the highest level CAPS mode enabled to complete the diagnosis.

The following command functions are not permitted within CAPS Recording areas.

BRANCH	CALL
ERRET	RETURN
COMP	CCOMP
MCOMP	DO loops containing TEST LGC command functions.

NOTE

Additional CAPS-related techniques and commands are described in Appendix C at the rear of this manual.

3.2.6 Logic Drivers and Sensors (LGC) [All Test Systems]

General

Logic tests are performed using two programmable logic sets (A and B). Each logic set includes high and low DUT input signals and high and low DUT output signals. A logic test program must include an ASSIGN LGC command which contains the required parameters that define the logic levels. This command must be executed before any other logic commands can be executed. Beyond this point in the program, all logic test functions refer to these logic levels.

A summary of logic test commands and parameters is provided on the following page. The remaining paragraphs describe the various logic test commands in detail and the procedure for writing logic test programs.

Logic Test Commands

```
ASSIGN LGC IA(2, 4, 6) VILA=.5 VIHA=4.5  
            IB(8, 10, 12) VILB=-4 VIHB=2.9  
            OA(1, 3, 5) VOLA=1.5 VOHA=3.9  
            OB(7, 9, 11) VOLB=-3 VOHB=1;
```

The above ASSIGN LGC statement defines the parameters to be used in logic tests which follow and supercedes all previous ASSIGN LGC commands. IA is DUT input logic level set A. Pins 2, 4, and 6 are assigned to this logic level set. The remainder of that line defines the actual voltage levels to be used; VILA (DUT voltage input-low A) is 0.5 volts; VIHA (DUT voltage input high A) is 4.5 volts. Similarly, the second line defines input logic level set B. The remaining two lines define DUT output pins and logic level sets and the actual voltages involved. In the statements which follow, levels are set or measured by stating IH (input high), IL (input low), OH (output high) and OL (output low) and including the pins involved with the test statement. The pins were assigned to a logic level set by the ASSIGN LGC command parameters. Hence, all levels presently assigned will remain assigned to the specified pins until the logic drivers and sensors are cleared, or superseded (by another ASSIGN LGC command).

Partial ASSIGN LGC commands (those that define a single logic set (A or B), or specify only input (I) or output (O) functions) are permitted. However, low and high voltage outputs (VOL(A/B) and VOH(A/B)) must always be specified for the logic set assigned. In addition, low and high voltage inputs VIL(A/B) and VIH(A/B) must be included if input pin functions (IH or IL) are specified.

NOTE

Although, in 1792 A/B Test Systems, the low and high logical levels are preset by hardware; the voltages must be included in the ASSIGN LGC command statement.

```
REASSIGN LGC I (7, 9, 11) O(8, 10, 12);
```

The REASSIGN LGC command is used to change assigned logic drivers to logic sensors and vice versa. The above example would reverse the input/output function specification of logic set B as initially assigned by the previous ASSIGN LGC example; unspecified pins remain unchanged (REASSIGN LGC commands can not change logic level sets and the actual voltages involved).

```
SET LGC IH(2, 4, 6) IL(8, 10, 12) OH(7) OL(9, 11);
```

The SET LGC command is used to activate logic drivers and to specify logic states for sensors. No test results are produced. In the above example, driver pins 2, 4, and 6 are set high (4.5 volts using the previous ASSIGN LGC example), and pins 8, 10, and 12 are driven low (-4 volts). Output pin 7 is specified as high, and 9 and 11 are low; however, no sensor test results are used.

```
$IH(2, 4, 6) IL(8, 10, 12) OH(7) OL(9, 11);
```

The SET LGC command can be replaced with the \$ symbol. The program then executes the command exactly as the SET LGC command. (Note that the command shown above and the SET LGC command above are equal.)

TEST LGC IH(2, 4, 6) IL(8, 10, 12) OH(7, 9, 11) OL(1, 3, 5) ITT=20M IST=5M [150];

The TEST LGC command uses DUT output logic levels previously established by executing an ASSIGN LGC command. When executed, the specified output pins are examined for the specified logic levels. When an output low (OL) parameter is stated, any measured DUT output *more negative* than the ASSIGNED value is considered a valid test result. Similarly, an output high (OH) parameter causes a valid test result to occur when measured DUT output pins specified are *more positive* than the ASSIGNED value. In the above example, specified output pins (1, 3, 5) which are *not* more negative than the ASSIGNED value and/or pins (7, 9, 11) not more positive than the ASSIGNED value result in an error, and the program branches to command 150.

2045: MCOMP LGC IH(2, 4) IL(6, 8) OH(5, 7) OL(1, 3) [2130];

This is a measure and compare command which operates exactly as a TEST LGC command, except that instead of flagging an error when the test conditions are not met, the program simply branches to the branch address parameter (required). In the above example, input conditions are generated as previously described. If the stated output conditions do not occur, the program branches to command 2130. An MCOMP command can be useful when it is desired to repeat an operation until certain conditions develop, such as cycling a counter until a starting count is attained.

```
DO 100;
MCOMP LGC IH(1) IL(2) OH(3) OL(4) [50];
ESCAPE 51;
50: DONE;
WRITE 'ERROR MESSAGE';
51: NOP;
```

In the above example, MCOMP is part of a DO loop. If the test conditions of the MCOMP command are met, the program branches out of the DO loop via the ESCAPE command. However, if the test conditions are not met, the program remains in the DO loop. In this example, the MCOMP command will be executed 100 times if the test conditions are not met. Then, the program will execute the WRITE command; following the DONE command, and display an error message, and then execute the command at 51. DO loop execution is discussed in para. 3.4.8.

IGNORE LGC (2, 13);

The IGNORE LGC command causes the test program to ignore specified terminals (2 and 13 in the above example) in the TEST LGC or MCOMP LGC pass-fail decision. The command remains effective until another IGNORE LGC command is executed. Both inputs and outputs can be ignored.

SET PULSE-8 2E2; or SET PULSE-8 200; or SET PULSE-8 REG-3;

The SET PULSE command enables a single logic pin driver to function as a pulse generator. Required parameters include the logic pin number (pin 8 in the above example) and a floating point value for the number of complete pulses. An assigned PNAME (Para. 3.4.15) cannot be used in place of the logic pin number in SET PULSE statements. The floating point value must be specified in any of three ways, as implied by the above example. The initial condition (baseline) of the pulse train is the currently programmed high or low logic state for the specified driver pin; logic levels are as previously defined by the most-recent ASSIGN LGC command. Each complete pulse requires two driver strobes (pulse transitions coincide with the strobes). The time between driver strobes is the currently active inter-test time (ITT). This timing can be specified in the SET PULSE command statement with the optional ITT=parameter. The optional IST=parameter may also be included in the statement, but has no effects on the operation of the SET PULSE command.

CLEAR LGC;

This command causes all drivers to disconnect (become inactive); all previously programmed logic commands are cleared. The CLEAR ALL command produces the same result in addition to clearing all test system functions.

System Timing (IST-ITT)

When the test application requires it, logic-test timing can be established through the use of the command parameters: IST (Inter-Strobe Time). These parameters are used in SET LGC and TEST LGC commands as follows:

TEST LGC IH (2, 4, 6) IL (8, 10, 12) OH (7, 9, 11) OL (1,3, 5) ITT = 20M IST = 5M;

When a TEST LGC command is executed, input-pin conditions are updated and driven in accordance with the command statement, and then the output pins are sensed. The elapsed time between driving the input pins and sensing the output pins is the Inter-Strobe Time (IST). IST can be specified up to a maximum of 1.6384 seconds in increments of 5.0 μ s (1792 A/B, 1795 and 1799 systems), or 100 ns (1792 C/D systems). Once specified, IST remains in effect for all subsequent logic tests until respecified or else cleared by an ASSIGN LGC, CLEAR ALL or BEGIN CAPS command. When not specified, actual IST defaults to the minimum achievable value permitted by the test system, as described below for each test system.

The elapsed time for completely executing a SET LGC or TEST LGC command statement (from driving the inputs of the previous command to driving the inputs for the current command) is Inter-Test Time (ITT). ITT can be specified up to a maximum of 16.384 seconds in increments of 1.0 μ s. Once specified, ITT remains in effect for all subsequent logic commands until respecified or cleared by an ASSIGN LGC command. When not specified, actual ITT defaults to the minimum achievable value permitted by the test system. Minimum ITT is a function of the test system type, the number of driver-sensor pins ASSIGNED, and the use of statement labels or IST specifications. Minimum ITT for a given system type is determined as shown in the following paragraphs.

1792 A/B

IST:

- 12.0 μ s (default)
- 1.6384 seconds (max)
- 5.0 μ s (min resolution increment)

ITT: — depends on the command and whether label processing is required. Read the cumulative factors for a given command type *down* the column to find the calculations for ITT. All times are approximate, and are in μ s unless otherwise stated.

	SET LGC (No Labels)	SET LGC (With Labels)	TEST LGC (No Labels)	TEST LGC (With Labels)
Load Drivers Time μ s/Bd	7.8n	7.8n	7.8n	7.8n
Default IST	none	none	12.0	12.0
Sensing Time	none	none	5.2n	5.2n
Comparison and Label Process Time	30	302m	77.7	353m
Software Overhead	none	5.0n	31.8n	31.8n
Total ITT =	7.8n + 30	12.8n + 302m	89.7 + 44.8n	12 + 44.8n + 353m

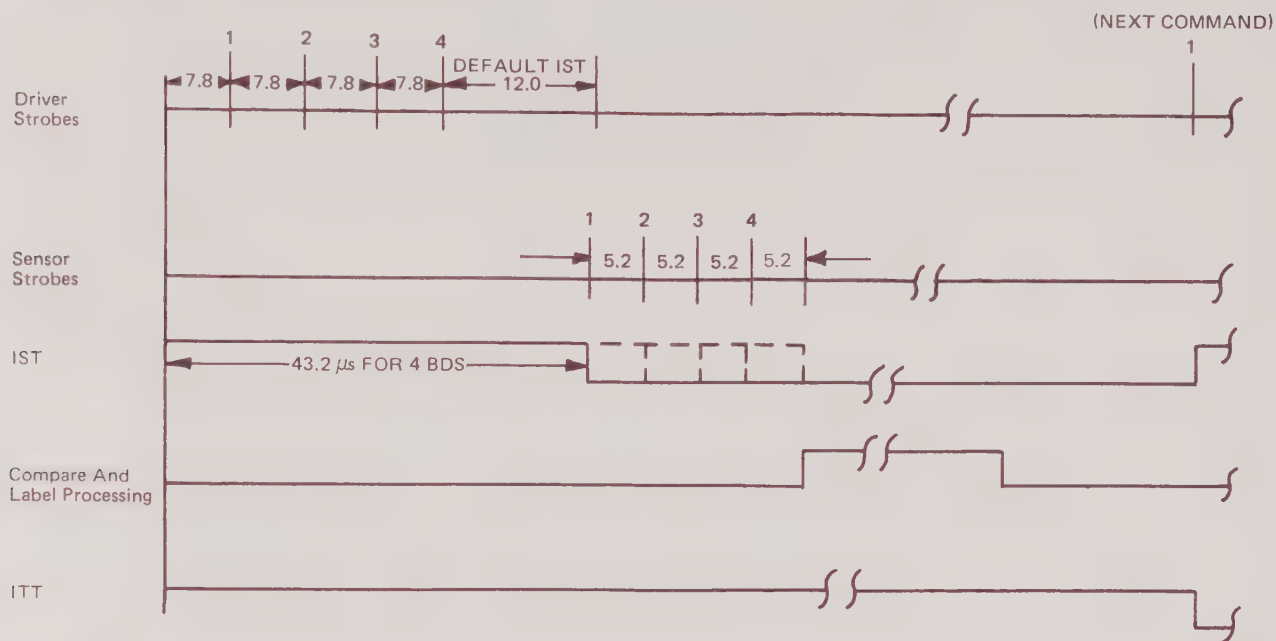
where:

m = the time for *each* label.

n = the highest driver or sensor pin number, divided by 12,
with remainders rounded out to the next highest number.

The ITT timing is illustrated below for a system containing four driver-sensor boards. In this example, assuming all driver-sensor pins have been ASSIGNED, IST can have any value specified in a test program command,

however, the actual minimum value for IST is $7.8 (4 \text{ Bds}) + 12.0 \mu\text{s} = 43.2 \mu\text{s}$. Note also, that the comparison time required when a TEST LGC is executed cannot begin until the sensor strobe time for *all* boards (cumulative) has completed.



1792 C/D

IST:

- 3.0 μs (default)
- 1.6384 seconds (max)
- 100 ns (min resolution increment)

ITT: — depends on the command and whether label processing is required. Read the cumulative factors for a given command type *down* the column to find the calculations for ITT. All times are in μs unless otherwise stated.

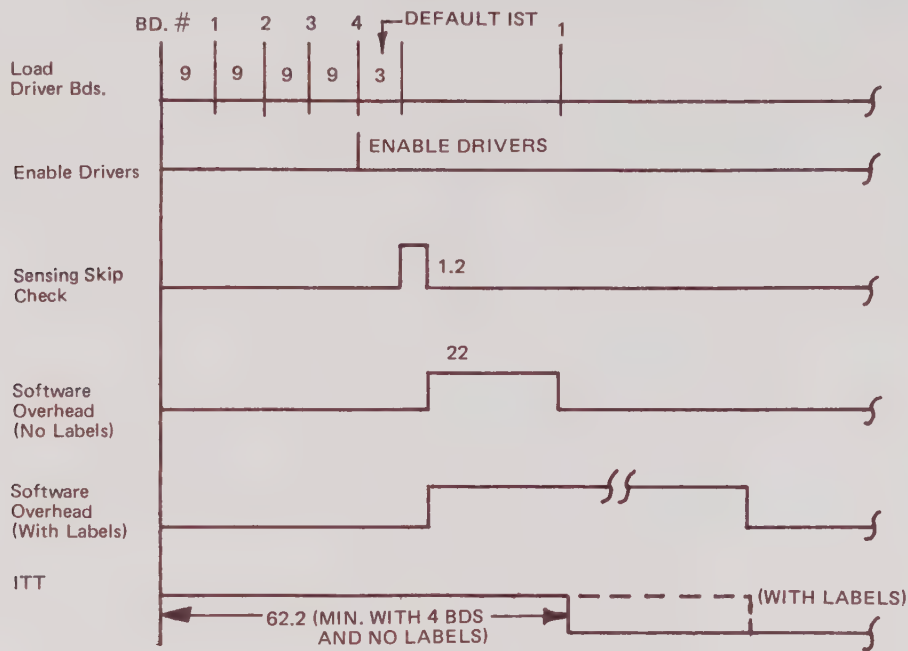
	SET LGC (No Labels)	SET LGC (With Labels)	TEST LGC (No Labels)	TEST LGC (With Labels)
Load Drivers Time μs/Bd	9n	9n	9n	9n
Default IST	3.0	3.0	3.0	3.0
Sensing Skip Check	1.2	1.2	1.2	1.2
Software Overhead and Label Process Time	22	220m	22	220m
Total ITT =	26.2 + 9n	4.2 + 9n + 220m	26.2 + 9n	4.2 + 9n + 220m

where: m = the time for *each* label.

n = Highest Input Driver Pin changed (HIDP) divided by 12 and rounded upward to nearest whole number, minus Lowest Input Driver Pin changed (LIDP) divided by 12 and rounded upward to nearest whole number, plus 1:

$$n1 = \left[\frac{\text{HIDP}}{12} \text{ (ROUNDED UP)} - \frac{\text{LIDP}}{12} \text{ (ROUNDED UP)} \right] + 1$$

The ITT timing is illustrated below for a system containing four driver-sensor boards. In this example, assuming all driver-sensor pins have been ASSIGNED, IST can have any value specified in a test program command, however, the actual minimum value IST will assume with four boards is $9(4 \text{ Bds}) + 3.0 = 39 \mu\text{s}$.



1795

IST: 15.0 μs (default)
 1.6384 seconds (max)
 5.0 μs (min resolution increment)

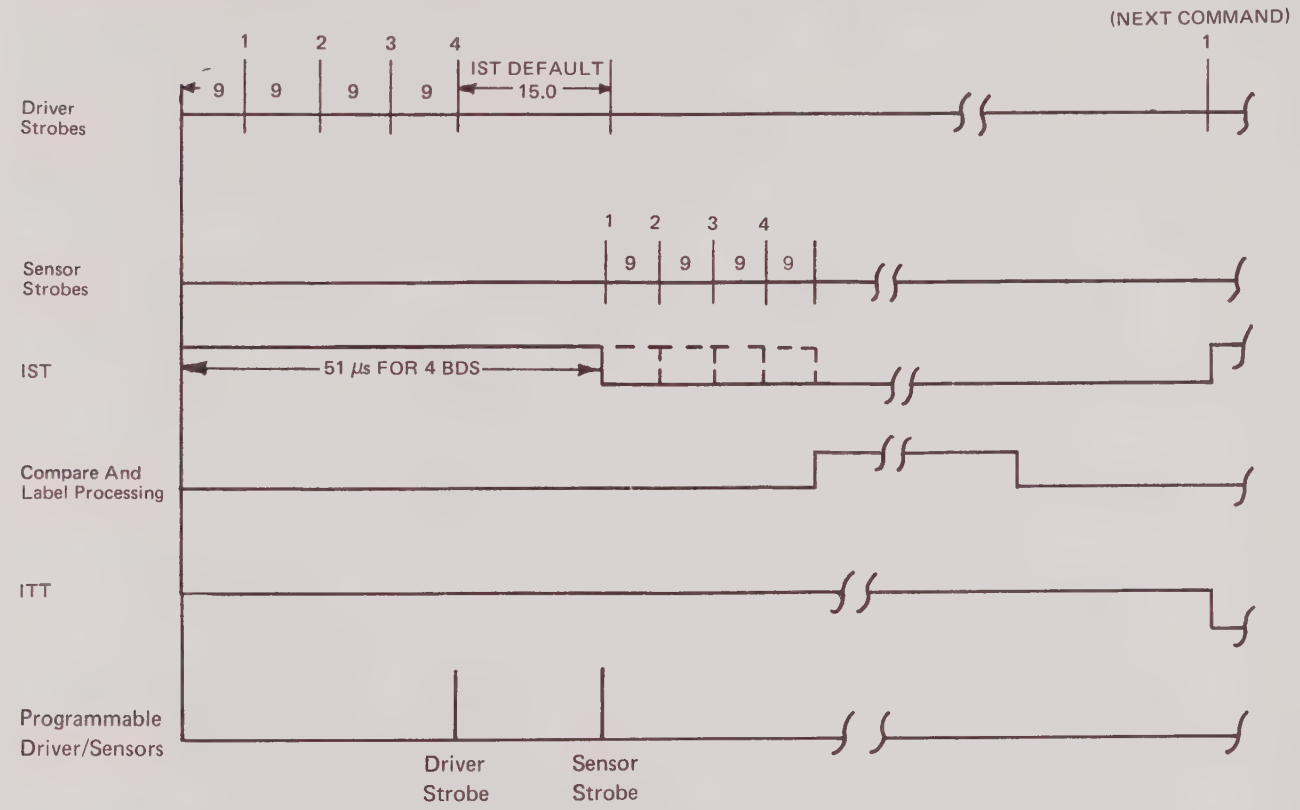
ITT: – depends on the command and whether label processing is required. Read the cumulative factors for a given command type *down* the column to find the calculations for ITT. All times are in μs unless otherwise stated.

	SET LGC (No Labels)	SET LGC (With Labels)	TEST LGC (No Labels)	TEST LGC (With Labels)
Load Drivers Time $\mu\text{s}/\text{Bd}$	9n	9n	9n	9n
Default IST	none	none	15.0	15.0
Sensing Time	none	none	9n	9n
Comparison and Label Process Time	39	364m	90	424m
Software Overhead	none	6n	42n	40n
Total ITT =	9n + 39	15n + 364m	105 + 60n	15 + 58n + 424m

where: m = the time for *each* label.

n = the highest driver or sensor pin number, divided by 12,
 with remainders rounded out to the next highest number.

The ITT timing is illustrated below for a system containing 4 standard driver-sensor boards. The driver and sensor strobes are incremental as shown. For systems with the optional programmable driver-sensors, all boards are driven and sensed in parallel. In the timing shown below, assuming all driver-sensor pins have been ASSIGNED, IST can have any value specified in a test program command, however, the actual minimum value for IST is $9(4 \text{ Bds}) + 15.0 \mu\text{s} = 51 \mu\text{s}$. Note also that the comparison time required when a TEST LGC is executed cannot begin until the sensor strobe time for **all** boards (cumulative) has been completed.



Logic Test Programming

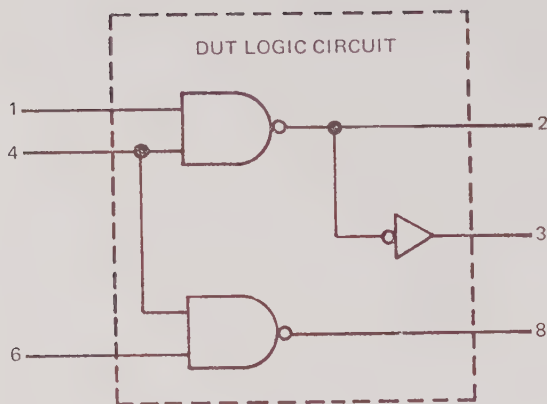
Logic Test programming is most easily described through the use of programming examples. Two examples are described below. The first example illustrates basic logic test programming, using a simple circuit. The second example is somewhat more complex, consisting of logic testing of a two-decade counter.

A Simple Example:

A typical test program (less power supply programming) for a simple logic circuit is shown below. The elements that make up any logic test program are input (I) and output (O) specification statements (ASSIGN LGC commands), a group of test statements (TEST LGC commands), and an END statement that signifies program completion. The decimal numbers contained in parentheses (pin list) represent driver/sensor (D/S) pin numbers. Note that these numbers are not usually the same as the DUT pin numbers. However, for the simple example included below, assume that the DUT device adaptor wiring provides the D/S pin number interface as shown.

NOTE

The use of the TEST LGC and SET LGC words in the following examples is not required. The Pnemonics TEST LGC need not be stated at all. A command with input and/or output specifications that is not preceded by anything implies TEST LGC. A "\$" can be substituted for the SET LGC wording.



Inputs	Outputs	Test No.
1 4 6	2 3 8	
0 0 0	1 0 1	10
0 0 1	1 0 1	20
0 1 0	1 0 1	30
0 1 1	1 0 0	40
1 0 0	1 0 1	50
1 0 1	1 0 1	60
1 1 0	0 1 1	70
1 1 1	0 1 0	80

DUT Truth Table

```

ASSIGN LGC IA(1, 4, 6) VIHA=3.5 VILA=.5
      OA(2, 3, 8) VOHA=3 VOLA=1;
10: TEST LGC IL(1, 4, 6) OL(3) OH(2, 8);
20: TEST LGC IH(6);
30: TEST LGC IL(6) IH(4);
40: TEST LGC IH(6) (OL(8);
50: TEST LGC IH(1) IL(4, 6) OH(8);
60: TEST LGC IH(6);
70: TEST LGC IL(6) IH(4) OL(2) OH(3);
80: TEST LGC IH(6) OL(8);
END;

```

The ASSIGN LGC statement defines the input and output pins for the test system. In addition, the statement establishes input and output logic voltage levels for logic set A (logic set B is not used in this example). All logic test programs require ASSIGN LGC statements for at least one logic set, as shown above. Inputs and outputs are indicated by the letters I and O, respectively.

The first line of the ASSIGN LGC statement defines input pins as follows: "Input" pins for logic set A are pins 1, 4, and 6. Voltage level for Input High for logic set A is 3.5 V. Voltage level for Input Low for logic set A is 0.5 V." The second line of the ASSIGN LGC statement defines output pins and voltage levels for logic set A in exactly the same manner as for the input pins.

TEST LGC commands specify the input conditions and output conditions expected during each test. The statements are incremental in that only those inputs and outputs that **change** state (with respect to previous TEST LGC commands) need be listed for each test. Immediately following the ASSIGN LGC statement, all inputs and outputs are assumed low until explicitly changed. As each test is executed by the test system, **all** inputs and outputs (whether specified or not) are checked. Thus, a specific input/output pattern **always exists** for **every test** whenever it is executed. Inputs and outputs not specified for a test are automatically added to the program during translation.

NOTE

The logic test programmer must be aware of two ordering schemes. One of these schemes, known as physical (or translation) ordering is the sequence in which statements appear in the source program. The other ordering scheme is the **execution order**, in which the statements are executed during the testing process. As an example, the physical order of the following program: 1, 2, 3, 4, 5, 6, 7. However, the execution order is 1, 2, 5, 6, 3, 4, 7.

/EXAMPLE 1/

```
1: ASSIGN LGC IA(1-12) VIHA=3.5 VILA=.5 VOHA=3.5 VOLA=.5;
2: BRANCH 5;
3: TEST LGC IH(1) IL(2-12);
4: BRANCH 7;
5: TEST LGC IH(2) IL(1, 3-12);
6: BRANCH 3;
7: END;
```

As an aid to programmers, logic test statements are analyzed in their translation order, and if the pin states do not change, their state may be omitted from the terminal specification. All pins are assumed initially low for this analysis. Using this aid to shorten the program, the example could now be written as:

/EXAMPLE 1/

```
1: ASSIGN LGC IA(1-12) VIHA=3.5 VILA=.5 VOHA=3.5 VOLA=.5;
2: BRANCH 5;
3: TEST LGC IH(1);
4: BRANCH 7;
5: TEST LGC IH(2) IL(1);
6: BRANCH 3;
7: END;
```

The END statement signifies program completion, and **must** be included at the end of every test program.

Certain test language features can be used to simplify the test program. Initially, all inputs and outputs in the test program are assumed **low**, the initial conditions are not altered until a TEST LGC command is executed which contains high-logic level parameters in the input or output specification. Hence, in the simple example shown, test 10 can be shortened to:

```
10: TEST LGC OH(2, 8);
```

Low inputs 1, 4 and 6 and low output 3 conditions are assumed by the test program since the initial state of all inputs and outputs is low. Note that the low state voltage level is as specified in the ASSIGN LGC commands which refer to each input and output used in the test program.

The next test is written as

```
20: TEST LGC IH(6);
```

because only inputs or outputs that **change** need be listed; in test 20 only input 6 changes.

Programming time can be shortened still further through the use of a sharp character (#). This character is used to invert the sense of a list. For example, the statement

```
TEST LGC IH(#1);
```

means set **all** inputs **except** 1 high and leave input 1 as previously set. Hence, in the simple example, test 50 can be written:

```
50: TEST LGC IH(1) IL(#1) OH(8);
```

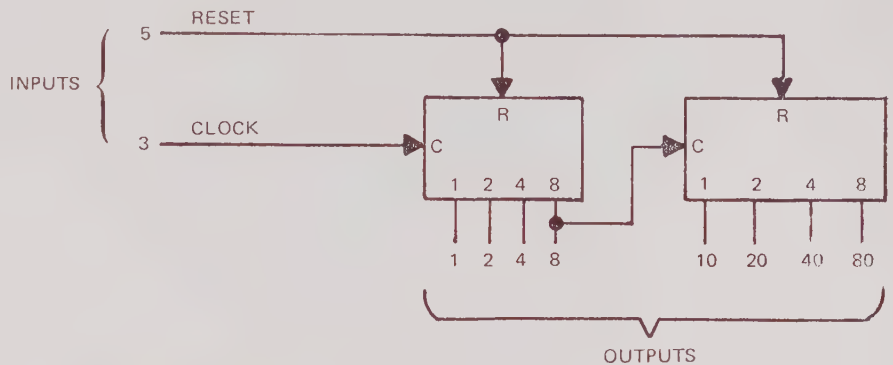
This would result in all inputs **except** input 1 being set low, which is the desired condition for test 50. If test 50 was written

```
TEST LGC IL(#1) OH(8);
```

all inputs would be set low.

Although sequential test numbers (labels) are used in the simple example, it is not a requirement of the test language that numbers in the label field to follow in sequence. When long programs are involved, it may be advantageous to skip test numbers at appropriate points in the test program so that program changes can be added without renumbering the entire test program. Any decimal numbers ranging from 1 to 8,388,607 can be used for test numbers. The only requirement is that test numbers cannot be duplicated.

A More Complex Example



```

ASSIGN LGC IA(3, 5) VIHA=3.5 VILA=.5
      OA(1, 2, 4, 8, 10, 20, 40, 80) VOHA=3.5 VOLA=.5;
10:  TEST LGC IH(5)           /RESET -- ALL OUTPUTS LOW/;
    TEST LGC IL(5)           /OUTPUTS REMAIN LOW AFTER RESET PULSE/;
30:  TEST LGC IH(3)           /COUNT TO 1/;
    TEST LGC IL(3) OH(1)     /COUNTER INCREMENTS ON TRAILING EDGE OF COUNT/;
    TEST LGC IH(3)           /COUNT TO 2/;
60:  TEST LGC IL(3) OL(1) OH(2);
    DO 2                      /COUNT TO 4/;
70:  SET LGC IH(3)           /COUNT PULSE/;
80:  SET LGC IL(3)           /INCREMENT COUNTER/;
    DONE;
90:  TEST LGC OL(#4) OH(4)    /CHECK THAT COUNT IS 4/;
100: DO 4                     /COUNT TO 8/;
110: SET LGC IH(3);
    SET LGC IL(3);
    DONE;
120: TEST LGC OH(8) OL(#8);
    DO 2                      /COUNT TO 10/;
130: SET LGC IH(3);
    SET LGC IL(3);
    DONE;
150: TEST LGC OH(10) OL(#10);
    DO 10                     /COUNT TO 20/;
160: SET LGC IH(3);
    SET LGC IL(3);
    DONE;
180: TEST LGC OH(20) OL(#20);
    DO 20                     /COUNT TO 40/;
190: SET LGC IH(3);
    SET LGC IL(3);
    DONE;
210: TEST LGC OH(40) OL(20);
    DO 40                     /COUNT TO 80/;

```

```

220: SET LGC IH(3);
    SET LGC IL(3);
    DONE;
240: TEST LGC OH(80) OL(40);
    DO 8                                /COUNT TO 88/;
    SET LGC IH(3);
    SET LGC IL(3);
    DONE;
270: TEST LGC OH(8);
    TEST LGC IH(5) OL(8)              /RESET — ALL OUTPUTS LOW/;
    TEST LGC IL(5)                    /OUTPUTS REMAIN LOW AFTER RESET PULSE/;
    DO 77;
300: SET LGC IH(3);
    SET LGC IL(3);
    DONE;
320: TEST LGC OH(8, 80);
330: TEST LGC IH(5) OL(8);
    TEST LGC IL(5);
    SET PULSE-3 100                   /COUNT AROUND FROM 00 TO 00/;
360: TEST LGC                         /TEST ALL OUTPUTS LOW/;
END;

```

Shown above is a simplified block diagram of a two-decade counter and the test program listing. The counter can record up to 99 input pulses. It has a reset input that forces the count to the 00 state and a clock input whose negative-going input transition increments the count by one. The output of each decade consists of four signal lines containing a BCD digit. A pulse applied when the counter is in the 9 state will reset the first decade to zero and increment the second decade. The counter will reset entirely to zero on a clock pulse following the 99 state. To prepare the program, first, analyze the circuit to determine the best method of testing. One approach for this type of device might include the following steps:

- a. Set the counter to 00 with the reset input and verify that all output bits are low.
- b. Count to 1, 2, 4, and 8 to verify that each bit of the units decade responds properly.
- c. Count to 10 to check that the carry between decades functions.
- d. Count to 20, 40, and 80 to test each bit of the tens decade.
- e. Count to 88 and then reset to check that the 8 and 80 bits will reset.
- f. Count to 77 and then reset to check that the remaining bits will reset.
- g. Count to 99 and then count once more to verify that the tens decade returns to zero.

Next, use test language statements to accomplish these steps. The test program listing shown will accomplish the above objectives. Note that the test program could include a comment statement at the top of the listing, such as:

```
/TEST PROGRAM FOR DUAL DECADE COUNTER/
```

Comment statements can be used in the program listing as an aid in understanding the program or for program diagnostic purposes.

The first non-comment statements are ASSIGN LGC input/output specification statements, similar to those included in the simple example. Note that the test numbers, in the label field, are in ascending order in increments of ten. This allows for easy program modification in the future without changing the ascending order and the presently used test numbers.

Since all inputs and outputs are initially low, it is only necessary to specify those inputs or outputs that should be high for the first test. This statement places a high on RESET input 5; all outputs should be low. The next statement returns input 5 to the low state and the outputs are again checked for low states. The next four test statements (30 through 60) alternately raise and lower the CLOCK input while checking the appropriate outputs at each test.

The next group of tests (70, 80 and 90) check for proper output on a count of 4. Since the counter is presently at a count of 2, it is only necessary to increment the counter twice. A DO loop is used for this purpose (see para. 3.4.8). Since it is not necessary to again check outputs 1 and 2 while counting to 4, SET LGC commands are used for incrementing the counter. The DO loop statements contained between the DO and DONE statements are executed two times, causing the counter to increment twice. Next, test statement 90 is executed, causing the test program to test for the proper counter outputs. The sharp character (#) is used to check that all outputs except output 4 are low and that output 4 is high.

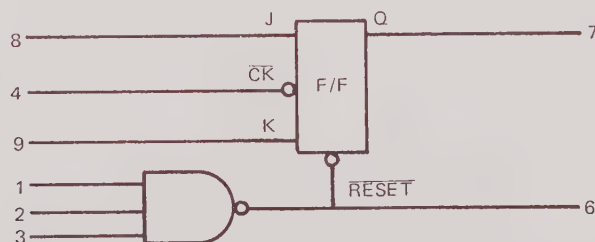
Successive DO loops set-up counts for checking outputs at counts of 8, 10, 20, 40, 80, and 88. After testing the count 88 (test 270), the RESET (input 5) is retested. Starting at the reset state, the counter is then incremented to a count of 77, as checked in test 320. The remaining counter bits are then checked for RESET operation in test 330. Finally, a complete counting cycle from 00 through 99 to 00 is checked in tests 350 through 360. The END statement signifies the end of the test program.

Optimizing Logic Tests

Logic test source programs can be optimized for minimum size and maximum execution speed by following four basic rules:

1. Consecutive logic commands must have common ITT and IST parameters (once specified, they remain until explicitly changed).
2. Branch addresses are not used.
3. Common commands are in sequence; include as many SET LGC commands or TEST LGC commands in sequence as possible.
4. Labels are not used.

A sample portion of an optimized test program is provided below. Note that the comments do not affect execution time. It is assumed that an ASSIGN LGC command has established test conditions, DUT power is applied, and that ITT and IST times are default values. The DUT logic circuit is also shown below:



Optimized	{	\$IH(1, 2, 3, 4, 8) IL(9) OH(6);	/SET LGC COMMAND RESETS FLOP OUTPUT FOR NEXT TEST/
		10: IL(1, 4) OH(7) [];	/TEST LGC COMMAND SETS FLOP. GATE IS HIGH/
		IL(2) IH(1);	
		IL(3) IH(2);	
		IH(3) OL(6, 7);	/RESET FLOP/
		IL(1, 8) OH(6);	
		IL(4);	/NO-TOGGLE TEST/
		IH(4, 8, 9);	
		IL(4) OH(7);	/TOGGLE HIGH/
		IH(4);	
		IL(4) OL(7);	/TOGGLE LOW/

NOTE

The first command in the optimized portions of the test program can include a label and a branch address. If a label and no branch address is included, the standard logic test error message will be displayed when the test conditions are not met. However, if a branch address is included, *any* test in the optimized portion which the DUT fails will cause the program to branch through the specified address, and the standard logic test error message will not be displayed.

3.2.7 Device Power Supplies (All Test Systems)

Device power supplies provide operating power (V_{CC}) for the DUT and are programmable to varying degrees depending upon the particular type of test system in use. Methods of programming each of the different types of 1792 test systems are described in the following paragraphs.

Note that test program statements prepared for one type of 1792 test system can be executed on any other 1792 test system, however, commands and arguments that do not apply will be ignored. In addition, some statements may function differently from system to system.

1792A Device Power Supplies

Each 1792A test system includes three *fixed* Device power supplies that are set by the system *hardware* (*not* test program statements) to provide the following power levels⁽¹⁾:

Power Supply	V	I
PS-1	+5V	3A
PS 2	+15V	1.25A
PS-3	-15V	1.25A

Test program statements are used in 1792A test systems to:

1. Simultaneously vary the power supply voltage outputs of all three power supplies by $\pm 5\%$ from normal levels⁽²⁾. The percentage offset is *fixed* by the system *hardware*, however, test program statements can be used to select either the fixed normal levels, fixed high levels (+5%), or fixed low levels (-5%).

2. Specify an absolute time for the power supplies to settle to the levels selected; otherwise, an error condition will occur and an error branch can be taken, if specified.

(1) Other levels and values are available on request. Fixed $\pm 10V$ or $\pm 12V$ power supplies may be substituted for the $\pm 15V$ supplies listed for PS-2 and PS-3 in some test systems, for example.

(2) Power supplies may be offset from normal by different percentages on request, $\pm 10\%$ limits may be established in some test systems, for example.

The test program statement needed to perform these functions in 1792A test systems may consist of the following labels, commands, and arguments:

123: SET PS-1 V=5.0 I=3 DLY=10M [210];

Where;

123:	is an optional label that may be included, if desired.
SET PS-n	is the Set <u>P</u> ower <u>S</u> upply command (1, 2, or 3 may be specified for n).
V=	is the <u>V</u> oltage specification for the power supply listed. This specification sets power supplies to normal, high, or low levels in conjunction with the DIAGNOSIS Mode SELECT MISCELLANEOUS OPTIONS Page VO= Option, and OPERATOR CONTROLS Page V= Option.
I=	is the Current specification for the power supply listed. The current specification must be listed as I=3 for PS-1 or I=1.25 for PS-2 and PS-3.
DLY=10M	is the time delay desired (absolute time for the power supply to settle). Up to a maximum of 511 seconds can be specified in 1 ms increments to cover the range from 1 ms (1M) to 511 seconds (511). If a DLY is not listed, the last previously specified DLY will apply. If the program does not contain a DLY then a default DLY of 23 ms will apply automatically. In the example shown, a delay of 10 ms (10M) has been listed.
[210]:	is the optional error branch address [210]. If the power supply does not settle within the maximum time allowed, the test program will branch to the address specified. The semi-colon (;) is the last entry and terminates the command string.

A SET PS command issued for **any** of the three power supplies (PS-1, PS-2, or PS-3) will set all three power supplies to normal, high, or low levels simultaneously. The range selected (normal, high, or low) is computed by taking the SET PS command V= value and then adding or subtracting the percentage voltage offset (DIAGNOSE Mode VO= option) from the programmed value (depending on whether the DIAGNOSE Mode V= option is set HIGH (add) or LOW (subtract). If the V= option is set to PROG, then the SET PS value will **not** be changed. The resulting value is then compared to the following set of voltage ranges to determine the actual voltage outputs of the power supplies:

<i>Device</i>	<i>Voltage Ranges</i>	<i>Actual Supply Voltages</i>	
PS-1 (+5V)	4.375 to <4.875	4.75	LOW
	4.875 to <5.125	5.00	NORMAL
	5.125 to <5.625	5.25	HIGH
PS-2 (+15V)	13.125 to <14.625	14.25	LOW
	14.625 to <15.375	15.00	NORMAL
	15.375 to <16.875	15.75	HIGH
PS-3 (-15V)	-13.125 to <-14.625	-14.25	LOW
	-14.625 to <-15.375	-15.00	NORMAL
	-15.375 to <-16.875	-15.75	HIGH

NOTE

Some 1792A test systems may have 10V or 12V supplies assigned to the PS-2 or PS-3 slots. The same voltage ranges listed for the $\pm 15V$ supplies are used to compute ranges for the 10V and 12V supplies, however, the actual supply voltages will be different.

All **three** power supplies are then set to the range selected (normal, HIGH or LOW) **simultaneously**. For example, the command

SET PS-1 V=5 I=3;

will set PS-1 to the normal (+5V) range and also set PS-2 and PS-3 to their normal ranges **if** the V= option is set to PROG. If the V= option is set to HIGH, **and** the VO= option is set to 5%, then the actual value used to select the range will be 5.25V (5.0+0.25) and all **three** power supplies will be set to their HIGH ranges.

The following additional Device power supply summary commands may also be used with 1792A test systems:

<i>Command</i>	<i>Meaning</i>
CLEAR PS;	This command turns off all 1792A Device power supplies simultaneously.
SET POWER OFF;	This command turns off all 1792A Device power supplies and keeps them off until a SET POWER ON command is executed (SET PS-n commands do not turn on power if they occur after a SET POWER OFF command and before a SET POWER ON command is executed).
SET POWER ON;	This command turns all 1792A Device power supplies on nearly simultaneously, depending on loads. The command removes the off states established by the SET POWER OFF command. Device power supplies are set to the last previously specified levels (if any) as the supplies are turned on.

NOTE

The SET POWER ON command is **not** required unless a SET POWER OFF command has been executed previously. Normally, SET PS-n commands turn Device power on.

SET POWER LMTCHK=OFF;	Adding the optional LMTCHK=OFF specification to the SET POWER command will disconnect the limit checking capability of the software, and power supply error messages will not be generated. The command is useful if it is felt that DUT loads might create an intentional limit condition which could cause the generation of error messages.
SET POWER LMTCHK=ON;	This command turns on the software Device power supply error checking capability previously disconnected by a LMTCHK=OFF command.

NOTE

LMTCHK=ON is assumed from program START until a LMTCHK=OFF command is specified. However, LMTCHK=OFF is assumed when using the CONTINUE ON ERROR Option of the DIAGNOSIS Mode

1792B Device Power Supplies

Each 1792B test system may include up to six Device power supplies furnished in the following configuration:

<i>Power Supply</i>	<i>Configuration</i>
PS-1	This is a single 3A power supply that can be programmed to operate over a –5V to +4.99V range in 10mV steps.
PS-2, PS-5, PS-6	Up to three 1A power supplies may be furnished. Each power supply can be programmed to operate over a –40V to +39.92V range in 80mV steps.
PS-3, PS-4	Up to two 2A power supplies may be furnished. Each power supply can be programmed to operate over a –20V to +19.96V range in 40mV steps.

Test program statements are used in 1792B test systems to:

1. Program each power supply to a specified voltage level within the voltage range of the supply involved and in the steps indicated.

2. Programmed voltage levels become the normal levels for the power supply involved, and may be raised or lowered as the need dictates. The DIAGNOSIS Mode SELECT MISCELLANEOUS OPTIONS Page VO= option, and OPERATOR CONTROLS Page V= option may also be used to set individual power supplies to levels above (HIGH) or below (LOW) the programmed normal levels by the percentage voltage offset specified by the VO= option. Power supplies cannot be set beyond the voltage ranges of each supply, however.

3. Specify a time delay so that the supply can settle prior to execution of the next statement.

The test program statement needed to perform these functions in 1792B test systems may consist of the following labels, commands, and arguments:

123: SET PS-1 V=3.0 I=3 DLY=10M;

Where:

123: is an optional label that may be included, if desired.

SET PS-n is the Set Power Supply command (1 through 6 may be specified for n).

V= is the programmed Voltage specification for the power supply listed. Program this supply to a level within the voltage range of the power supply in the steps indicated. The following voltage ranges and steps are available:

<i>Power Supply</i>	<i>Voltage Range</i>	<i>Steps</i>
PS-1	– 5.000 to +4.990V	10mV
PS-2	–40.000 to +39.920V	80mV
PS-3	–20.000 to +19.960V	40mV
PS-4	–20.000 to +19.960V	40mV
PS-5	–40.000 to +39.920V	80mV
PS-6	–40.000 to +39.920V	80mV

I=n is the current specification for the power supply listed. The current specification (n) **must** be listed as I=3 for PS-1, I=1 for PS-2, PS-4, and PS-5, and I=2 for PS-3 and PS-4.

DLY=10M;

is a time delay to allow the power supply to settle prior to executing the next statement. Up to a maximum of 511 seconds can be specified in 1 ms increments to cover the range from 1 ms (1M) to 511 seconds (511). The semi-colon (;) is the last entry and terminates the command string.

Note that an error branching argument is not listed as part of the command string in 1792B test systems, as the power supply current limiting capability is fixed by the system hardware and thus operates independently without reference to the software.

Some examples of 1792B SET PS statements might include:

<i>Command</i>	<i>Meaning</i>
SET PS-3 V=8.40 I=2;	Set PS-3 to +8.40V.
SET PS-3 V=-20 I=2 DLY=20M;	Set PS 3 to -20V. Allow 20 ms for the power supply to settle before executing the next statement.
SET PS-3 V=-20 I=2;	Set PS-3 to -20V.
SET PS-1 V=-3.0 I=3;	Set PS-1 to -3.0V.

The following additional Device power supply summary commands may also be used with 1792B test systems:

<i>Command</i>	<i>Meaning</i>
CLEAR PS-n;	This command programs the <i>designated</i> Device power supply to <i>zero</i> (1 through 6 may be specified for n).
CLEAR PS;	This command programs <i>all</i> 1792B Device power supplies to <i>zero</i> .
SET POWER OFF;	This command programs <i>all</i> 1792B Device power supplies to <i>zero</i> and keeps them at zero <i>until</i> a SET POWER ON command is executed (SET PS-n commands do not turn on power if they occur <i>after</i> a SET POWER OFF command and <i>before</i> a SET POWER ON command is executed).
SET POWER ON;	This command sets <i>all</i> 1792B Device power supplies to the <i>last</i> previously specified levels (if any). The command <i>removes</i> the zero states established by the SET POWER OFF command and allows any SET PS-n commands that come <i>after</i> the SET POWER ON command to function normally. Optional LMTCHK=OFF specifications do <i>not</i> apply in 1792B test systems.

NOTE

The SET POWER ON command is *not* required unless a SET POWER OFF command has been executed previously. Normally, SET PS-n commands turn Device power on.

1792C/D Device Power Supplies

Each 1792C test system may include up to six plug-in Device power supplies while each 1792D test system may have up to nine of the power supplies. Device power supplies plug into UPS boxes and each of the power supplies (with the single exception of the 80V power supply) can be inserted into *any* of the three slots available within each UPS box. The 80V power supply occupies two slots. PS-n slot numbers (where n may be 1 through 9) are assigned for programming purposes according to the power supply locations within each UPS box. Slot assignments together with power supply scanning codes are defined in Section 2 of the System Operating Instructions

The following Device power supplies may be furnished:

Type	Configuration
10V	This power supply can be programmed as a voltage source to operate over a -10.24 to $+10.235$ V range in 5mV steps. Current limits are also programmable over the range of -4.096 to $+4.088$ A in 8mA steps.
20V	This power supply can be programmed as a voltage source to operate over a -20.48 to $+20.47$ V range in 10mV steps. Current limits are also programmable over the range of -2.044 to $+2.048$ A in 4mA steps.
40V	This power supply can be programmed as a voltage source to operate over a -40.96 to $+40.96$ V range in 20mV steps. Current limits are also programmable over the range of -1.024 to $+1.022$ A in 2mA steps.
80V	This power supply is the only one that can be programmed to operate as either a programmable voltage source with programmable current limits (VL Mode) or as a programmable current source with programmable voltage limits (IL Mode). All of the other power supplies operate in the VL Mode <i>only</i> . The power supply is programmable over a voltage range of -81.92 to $+81.88$ V in 40mV steps, and over a current range of -0.512 to 0.51175 A in 0.25mA steps.

For proper operation, it should be noted that the 10V, 20V and 40V supplies can float at up to ± 4 V (dc or peak ac) from ground. (The maximum voltage between the low terminal and system ground for the power supply to be within specifications is ± 4 V.) The low terminal of the 80V supply can float as much as ± 250 V (dc or peak ac) from the system ground.

Test program statements are used in 1792C/D test systems to:

1. Program each power supply to specified voltage and current levels within the range of the supply involved and in the steps indicated
2. Programmed voltage levels become the normal levels for the power supply involved, and may be raised or lowered as the need dictates. The DIAGNOSIS Mode SELECT MISCELLANEOUS OPTIONS Page VO= Option, and OPERATOR CONTROLS Page V= Option may also be used to set *all* power supplies to levels above (HIGH) or below (LOW) the individually programmed normal levels by the percentage voltage offset specified by the VO= Option. Power supplies cannot be set above or below the supply limits, however.
3. List the time allowed for the power supplies to settle to the programmed levels; Otherwise, an error condition will occur and an error branch can be taken, if specified.
4. Program the 80V power supply to operate in either a Voltage Locked (VL) Mode where the supply operates as a programmable voltage source with programmable current limits, or in a current locked (IL) mode where the supply operates as a programmable current source with programmable voltage limits. All other Device power supplies operate in the VL Mode *only*.
5. Connect the power supply involved to the DUT via the power supply scanner system, if desired.

The test program statement needed to perform these functions in 1792C/D test systems may consist of the following labels, commands, and arguments:

123: SET PS-1 AT(1:4) VL V=4.8 I=100M DLY=10M [350];

Where:

123: is an optional label that may be included, if desired.

SET PS-n is the SET Power Supply command (1 through 9 may be specified for n).

AT(1:4) is the set scanner specification. In the example shown, the Device power supply primary output terminal is connected to shelf pin 1 and the power supply return is connected to shelf pin 4. Terminal designations are further defined in Para. 3.2.4. In addition, Para. 4.2 lists some of the program considerations required when scanning. Scanning codes are listed in the System Operating Instructions

VL and IL VL is an optional specification that establishes the Voltage Locked Mode. If *not* listed, VL is *assumed*. The VL Mode is used for all power supplies except the 80V supply. Either the VL or IL (Current locked) modes may be specified for the 80V power supply.

V= is the Voltage specification for the power supply listed. Program this supply to a level within the voltage range of the power supply in the steps indicated. The following voltage ranges and steps are available:

<i>Power Supply</i>	<i>Voltage Range</i>	<i>Steps</i>
10V	–10.24 to +10.235V	5mV
20V	–20.48 to +20.47V	10mV
40V	–40.96 to +40.96V	20mV
80V	–81.92 to +81.88V	40mV

I= is the current specification for the power supply listed. Program this supply for a current limit (or current source in the case of the 80V supply) within the current range listed in the steps indicated. The following current ranges and steps are available:

<i>Power Supply</i>	<i>Current Range</i>	<i>Steps</i>
10V	–4.096 to +4.088A	8mA
20V	–2.044 to 2.048A	4mA
40V	–1.024 to +1.022A	2mA
80V	–0.512 to +0.51175A	0.25mA

DLY=10M is the maximum time delay desired for the power supply to settle to programmed levels. If the power supply settles in less time, the full delay will *not* be executed. Up to a maximum of 511 seconds can be specified in 1 ms increments to cover the range from 1 ms (1M) to 511 seconds (511). If a DLY is *not* listed, a *default* DLY of 50 ms will apply automatically, regardless of any previous DLY specifications in the test program.

[350]; is the optional error branch address [350]. If the power supply does *not* settle within the maximum time allowed the test program will branch to the address specified. The semi-colon (;) is the last entry and terminates the command string.

The following CAUTIONS and NOTES apply when programming any of the 1792C/D Device power supplies:

CAUTION

Power supply scanner contacts are rated at 1A. Refer to the program considerations in Section 4 for details.

NOTE

When a SET PS statement including a decreasing (in magnitude) dependent variable (limit) is executed, the power supply output voltage and current goes from the previously programmed value to **zero** and then up to the new programmed level. For example:

When the power supply is in the VL mode, I is the dependent variable. When in the IL mode V is the dependent variable. Hence, assuming the VL mode executing the command sequence.

```
SET PS-1 V=10 I=1;  
SET PS 1 V=10 I=.5;
```

causes the power supply output to go from 10 V to 0 V to 10 V, with a new current limit of .5A (the current limit is the decreasing dependent variable). If the above commands were followed with

```
SET PS-1 V=10 I=1;
```

the current limit would be **increased** without changing the power supply output.

The following additional Device power supply summary commands may also be used with 1792C/D test systems:

<i>Command</i>	<i>Meaning</i>
CLEAR PS-n;	This command programs the designated power supply to zero and then turns off the supply. It also disconnects the power supply from the DUT if power supply scanning is being used. (1 through 9 may be specified for n).
CLEAR PS;	This command programs all 1792C/D Device power supplies to zero and then turns them off. It also disconnects all power supplies from the DUT when scanning.
SET POWER OFF;	This command programs all 1792C/D Device power supplies to zero and keeps them at zero until a SET POWER ON command is executed (SET PS-n commands do not turn on power if they occur after a SET POWER OFF command and before a SET POWER ON command is executed).
SET POWER ON;	This command sets all 1792C/D Device power supplies to the last previously specified levels (if any). The command removes the zero states established by the SET POWER OFF command. If different DLY specifications have been listed for different supplies, the largest DLY specification will apply. The command connects scanned pins, and allows any SET PS-n commands that come after the SET POWER ON command to function normally.

NOTE

The SET POWER ON command is *not* required unless a SET POWER OFF command has been executed previously. Normally, SET PS-n commands turn Device power on.

- SET POWER LMTCHK=OFF; Adding the optional LMTCHK=OFF specification to the SET POWER ON command will disconnect the limit checking capability of the software, and power supply error messages will *not* be generated. The command is useful if it is felt that DUT loads might create an intentional limit condition which could cause the generation of error messages.
- SET POWER LMTCHK=ON; This command turns *on* the software Device power supply error checking capability previously disconnected by a LMTCHK=OFF command.

NOTE

LMTCHK=ON is assumed from program START until a LMTCHK=OFF command is specified. However, LMTCHK=OFF is assumed when using the CONTINUE ON ERROR Option of the DIAGNOSIS Mode.

In 1792 C/D test systems, power supply load current and output voltage can be measured and tested as follows:

- 125: TEST PSI-1 <HI=.1 LO=.01> [345];
234: MEAS PSI-1 [445];

The TEST PSI and MEAS PSI commands measure the power supply load current. Either command must be preceded by a valid SET PS-n command, where n is the same power supply number used in the TEST PSI and/or MEAS PSI commands, placing the power supply in the voltage locked mode (VL). When executing the TEST PSI command, the power supply current is measured and compared to the limits given. If the test conditions are not met, program control is transferred to the command at the specified branch address and an error condition occurs. If no branch address is supplied, the test program executes a standard error handling routine. Both MEAS and TEST PSI place the measured value in the PSI register (via the AR register) for use by the test program.

NOTE

With MEAS and TEST PSI commands, the current limit for that supply is temporarily disabled, resulting in the possibility that the supply could reach its maximum current capability.

126: MEAS PSV-2 [450];

127: TEST PSV-2 <HI=10 LO=9.9> [350];

The TEST PSV and MEAS PSV commands measure the output voltage of the 80V power supply (PS-2 in this example) operating in the current locked mode (IL). Either command must be preceded by a valid SET PS-n command, placing the 80V supply in the IL mode. When executing the TEST PSV command, the power supply voltage is compared to the limits given. If the test conditions are not met, program control is transferred to the command at the specified branch address and an error condition occurs. Both MEAS and TEST PSV place the measured value in the PSV register (via the AR register) for use by the test program.

NOTE

MEAS PSI and MEAS PSV statements may include error-branch addresses as shown above. The error-branch address is taken when one or more of the following conditions exists.

1. Referenced PS is turned off,
2. Attempting to measure voltage when PS is in voltage-locked mode,
3. Attempting to measure current when PS is in current-locked mode,
4. Referenced PS is unable to supply the power specified by the V= and I= parameters, and LMTCHK=ON.

3.2.8 Sync Pulse Generation [All Test Systems]

SET SYNC statements can be used to generate positive- and negative-going trigger pulses* over one of six sync pulse lines. The statement has the form:

SET SYNC-n;

Where n is a number from 1 to 6 that indicates the line over which the pulse is to occur. Pulse lines 1 through 6 are available at interface connector jack J23 and can be used to trigger DUT circuits or circuits built into the device adaptor. In addition, positive-going pulses can be generated over the SYNC 1 BNC jack and negative-going pulses over the SYNC 4 BNC jack in the device adaptor well, and used to trigger external counters, oscilloscopes, generators, and other devices.

All SYNC 1 through 3 pulses are positive-going pulses (GND to +5 V) with a duration of 1 μ s.

All SYNC 4 through 6 pulses are negative-going pulses (+5 V to GND) with a duration of 1 μ s.

Multiple pulses can be programmed by appending an optional parameter. This parameter specifies the number of 1 μ sec. pulses to be generated and can be a floating-point number or a reference to a register containing the value. For example,

SET SYNC-2 25;

generates 25 positive-going pulses at the SYNC 2 pin on connector J23. If the parameter is not included, the program assumes that a single pulse is to be generated.

The repetition rate of SYNC pulse generation is higher than the repetition rate for normal test command execution. The pulses are produced at a rate greater than 100 kHz.

*Sync pulses are TTL-compatible logic signals. The 0 and +5 V levels are *nominal* logic levels and actual pulse amplitude will actually be less than the specified values.

3.2.9 Relay Drivers (RLY)

The relay drivers provide programmable drive signals (100 mA, 25V max.) to up to 24 external relay (or lamp) loads. Switching is accomplished by open-collector transistor drivers which sink load current via the external loads. Refer to the Operating Instructions for instructions in using the relay drivers. Two basic commands are used when programming the relay drivers: SET and CLEAR. The SET command requires O (open) and C (close) parameters which define the state of the drivers upon execution of the command, as shown in the following command:

```
SET RLY O(1, 3, 5, 7, 9, 11, 13, 15) C(2, 4);
```

In the above example, the SET RLY command causes drivers 1, 3, 5, 7, 9, 11, 13 and 15 to turn off and de-energize their associated relays, and drivers 2 and 4 to turn on, energizing their associated relays. The drivers not included in the above statement remain unchanged from the previously executed relay driver command (or CLEAR ALL command). Any required delay following the relay closure must be specified by a DELAY command.

```
CLEAR RLY;
```

Causes all relay drivers to become inactive (turn off), deenergizing their associated relay loads. The CLEAR ALL command produces the same result.

3.2.10 DC Voltmeter (DCV) [1792 D Option]

The DC voltmeter function is provided by the DC-to-digital converter. Programming the DC voltmeter is shown in the following examples:

```
290: TEST DCV AT (7:3) <HI=2.7 LO=2.3> [330] DLY=1M RNG=2 FORCE;
```

The DC voltage between terminal 7 and terminal 3 is measured, stored in the DCV register (via the AR register) and compared to the limits given. If the test conditions are not satisfied, the program branches to command 330. An optional statement in the parameter field DLY=n will cause a delay of n seconds (1023sec.max) between connection to the DUT and reading the measured value; the programmed delay allows settling time when it is known that the DUT circuit requires it. In addition, an optional range specification may be included. Range is specified by the statement RNG=n; immediately following the equal sign, the range is indicated by a code number. Use of the proper range in this specification reduces execution time of the command. If the range specified is incorrect, automatic ranging results (with specified DLY executed between each range change). However, if desired, automatic ranging can be suppressed by including the FORCE parameter, as shown. If FORCE is specified and the measurement is not on the range specified, a test error occurs, and no ranging takes place.

Code No.	Max DC voltage (range)
1	200.0 volts
2	20.0
3	2.0
4	0.2

```
170: MEAS DCV AT (7:3) RNG=3;
```

The DC voltage between terminal 7 and terminal 3 is measured and stored in floating point form via the AR register to the DCV register. The optional DLY, FORCE, branch address, and RNG parameters may also be included in a MEAS command, as described for the TEST DCV command. The branch address allows program branching if a measurement cannot be made.

```
120: CLEAR DCV;
```

This command causes the DC voltmeter to become disconnected from the DUT. (The CLEAR ALL command will also produce the same result with the DC voltmeter.) It causes the DC-to-digital converter to become disconnected from the DUT.

3.2.11 DC Current Meter (DCI) [1792 D Option]

The DC current meter function is also provided by the DC-to-digital converter. Programming the DC current meter is shown in the following examples:

540: TEST DCI AT (7:13) <HI=.1 LO=.05> [510];

The DC current meter is connected from terminal 7 to terminal 13. The current flowing from terminals 7 to 13 is measured, stored in the DCI register via the AR register, and compared to the specified limits. If the test conditions are not met, the program branches to command 510. The optional RNG, FORCE, and DLY parameters may be used with DCI commands, as described for the DC voltmeter. Ranges are specified below:

Code No.	Max DC current (range)	Current Sensing Ω
1	200 mA	1 Ω
2	20 mA	10 Ω
3	2 mA	100 Ω
4	200 μ A	1 k Ω
5	20 μ A	10 k Ω
6	2 μ A	100 k Ω

The total resistance in series with the measurement is the sum of the current-sensing range resistor (listed above) and the path resistance (scanner contact and cable resistance); typically $\simeq 1\Omega$.

MEAS DCI AT(7:13) [510];

The DC current meter is connected between terminals 7 and 13. The current flowing from 7 to 13 is measured and the value in amperes is placed in the DCI register via the AR register. The optional RNG, FORCE, branch address, and DLY parameters may be included as described for the TEST DCI command. The branch address allows program branching if a measurement cannot be made.

770: CLEAR DCI;

This command causes the DC current meter to become disconnected from the DUT. (The CLEAR ALL command will also produce the same result with the DC current meter.) It causes the DC-to-digital converter to become disconnected from the DUT.

3.2.12 AC Voltmeter (ACV)

The AC voltmeter function is provided by the AC-DC converter. AC voltage measurements are made with respect to ground; hence, a single terminal is specified. Programming the AC voltmeter is shown in the following examples:

140: TEST ACV AT (13) <HI=2.3 LO=1.9> SLOW [170];

The AC voltage is measured, stored in the ACV register (via the AR register), and compared to the limits specified. If the test conditions are not satisfied, the program branches to command 170. The optional SLOW parameter sets the AC voltmeter's time constant for making a measurement at a frequency less than 150 Hz. The optional LOWZ parameter is recommended for AC voltage measurements of signal frequencies greater than 150 kHz; this causes a low-impedance (50-ohm) termination to be switched in at the input of the AC voltmeter. In addition, the optional RNG, FORCE, and DLY parameters may be included, as described for the DC voltmeter (para. 3.2.10). Ranges are specified below:

Code No.	Max AC voltage (range)
1	150 volts rms
2	15 volts rms
3	1.5 volts rms
4	0.15 volts rms
5	0.015 volts rms

170: MEAS ACV AT (3) RNG=2;

The AC voltmeter is connected between terminal 3 and analog ground, and the AC voltage (rms) is measured. The result is stored in floating point form, in volts rms, in a register called ACV (via the AR register). Optional RNG, FORCE, DLY, LOWZ, SLOW and branch address parameters may also be included, as described for the TEST ACV command. The branch address allows program branching if a measurement cannot be made.

240: CLEAR ACV;

This command causes the AC voltmeter to become disconnected from the DUT. (The CLEAR ALL command will also produce the same result with the AC voltmeter.) It causes the DC-to-digital converter and AC-DC converter to become disconnected from the DUT.

3.2.13 AC Source (ACS)

The AC source function is provided by the function generator. It is capable of generating sine, square and triangular waveshapes over a wide range of amplitude and frequency. The AC source has a single output which is referenced to analog ground; hence, a single terminal is specified. Programming the ACS is shown below:

245: SET ACS AT (13) F=100 V=5;

The AC source high output is connected to terminal 13; the low output is connected to ground. Two required parameters follow the terminal specification which specify the desired frequency and amplitude which is to be generated. F defines the operating frequency in Hz. V defines the output pk-pk voltage into a 50 Ω load; open-circuit output voltage will be double the specified value. When no optional parameters are included, the test program assumes certain default conditions, including sine waveshape, continuous mode, and no DC offset voltage. Parameters are listed below

Two models of function generators, the WaveTek 154 and 157 are available. The programming statements are the same for each unit (except for dc mode). The programmable parameters for each model are listed below. The sources are capable of generating sine, square, and triangular waveshapes over a wide range of frequency and amplitude. The ACS has a single output which is referenced to analog ground. All references to the dc mode apply **only** to the Model 154.

Parameter	Model 154	Model 157
F= (freq. in Hz)	.001 — 9.9 MEG*	.000001 — 9.9 MEG
V= (volts ampl.)	.01 — 9.9 (pk-pk)	.001 — 9.9 V (pk-pk).
Optional Parameters:		
Output Signal:	TRI — triangular	
	RECT — rectangular	
	DC — DC voltage -- no AC component	
	(default) — sine	
Mode:	GATE — gates	
	TRIG — triggered	
	NO — no output	
	VCG — Voltage Controlled Frequency Generation	
	(for use with non-standard AC source)	
	(default) — continuous (normal mode)	
DC voltage offset:	OFF — volts DC offset. Range: ± 7.5 V including AC signal component.	
	(default) — 0 V offset	

Note that if the DC output signal parameter is specified, the output signal is simply a DC voltage, ranging from ± 0.1 —9.99 V. The DC voltage source impedance is 1 Ω . Note also that the programming resolution of voltage and frequency is to three decimal places. The gated and triggered modes require additional signal inputs to the ACS; refer to the ACS manufacturer's instruction manual for the Model 154 Programmable Waveform Generator (Wavetek) for the required connections and use of these modes.

CLEAR ACS;

This command causes the AC source to become disconnected from the source scanner and to produce no signal output. The CLEAR ALL command will also produce the same result with the AC source.

*not required if optional dc parameter is specified.

For systems equipped with two or three function generators, each AC source can be individually SET and CLEARED by adding qualifiers to the command as follows:

```
SET ACS-i AT . . . ;  
CLEAR ACS-i;  
where i = 1, 2, or 3.
```

In these multi-AC source systems, an unqualified SET ACS command defaults to ACS-1, and an unqualified CLEAR ACS command clears all AC sources.

3.2.14 Frequency Measurement Unit (FMU)

The Frequency Measurement Unit (FMU) is a very versatile unit capable of measuring the frequency (and therefore the period) of a signal, the time between two events, the ratio of two frequencies, the cumulative number of events (counts) during a time interval, etc. Because the events being measured are often the result of signals produced under program control, it is necessary in the program to 'set up' the FMU for the desired measurement, call for the desired stimulus signals, and then call for the measurement or test of the specified parameter.

The FMU includes two principal registers, SIG and CLK, and two input channels, A and B. Measurements are made by accumulating counts in these registers and then performing appropriate arithmetic operations in the computer.

The FMU is set up for any type of measurement by the SET FMU command, an example of which is shown below:

```
0173: SET FMU AT (3:24) TRIGA=2.4+ TRIGB=1.2-;
```

This command causes input Channel A to be connected to terminal 3 and input Channel B to be connected to terminal 24. The TRIGA parameter is the voltage threshold level of the signal event at the A input; the polarity sign that follows indicates the signal slope which will trigger the start of the measurement. Similarly, TRIGB is a parameter associated with the B input that it uses to stop the measurement, as applicable, when the second input channel is required. The TRIGA and TRIGB parameter values can be represented by referencing the contents of a register. For example: TRIGA=REG-1+. All other parameters associated with the FMU are optional, and include: PRECISION with which the measurement is to be made, CLK (clock), TRACKING mode, INTERVAL, FREQUENCY, FRATIO (frequency ratio), and COUNT, as described below.

TR in the parameter field sets the FMU in tracking mode, where it continuously makes the measurement it has been set up to make; the FMU can be interrogated at a later time by a MEAS or TEST command to determine the most recently measured value. The FMU need not be SET again for further interrogation when using the tracking mode.

INT (interval) must be present in the parameter field if the FMU is being set up for a TIME measurement; otherwise, it must not be present. The effect of INT is that when it is present, inputs A and B respectively start and stop the interval period.

ADIRect and BDIRect are optional parameters that correspond to input channels A and B respectively. An ADIR or BDIR specification causes a normally inserted 10:1 attenuator to be bypassed at the corresponding input channel. The ADIR and BDIR specifications are operational only if the associated TRIGA/TRIGB voltage specification is less than 3.0 volts; otherwise the ADIR and BDIR specifications are ignored and the 10:1 attenuator remains in the input circuit.

TIN is an optional terminate input parameter. When this parameter is inserted, *both* the A and B input channels are terminated with 50 Ω . Either channel can be independently terminated by specifying TINA or TINB, for channels A and B, respectively. Channels should be terminated when the measurement is from a low output impedance. Only 1 of the TIN, TINA, or TINB parameters can be specified at one time.

PREC= This is the PRECISION with which the measurement is to be made. Any PREC parameter integer from 1 to 8 can be specified except as indicated in the restriction table at the end of this section. The least precision that is acceptable, for the test at hand should be called for to avoid unnecessary test program execution time. If the PREC is not specified a precision of 5 (1 part in 10^6) is assumed.

PREC=	Precision
1	1 part in 10^2
2	1 part in 10^3
3	1 part in 10^4
4	1 part in 10^5
5	1 part in 10^6
6	Count mode only
7	10^6 Period
8	1 Period

CLK= The CLoCK= parameter is an integer code number specifying the frequency of the clock-count time base. Refer to the restrictions table at the end of this section for the clock conditions for any of the FMU operating modes.

Note that for frequency and period measurements that CLK can be 1, 2, or 3. CLK=1 is preferred whenever the expected frequency is above 1.2 Hz (or period below 830 ms). When measuring low frequencies between 0.12 Hz and 1.2 Hz (830 ms to 8.3 period) use CLK=2; CLK=3 is used for frequencies below 0.12 Hz (period longer than 8.3 s).

The default CLK Parameter if it is not specified is CLK=1 (10 MHz).

CLK=	Input to CLK
1	10 MHz
2	1 MHz
3	100 kHz
4	(do not use)
5	Input A
6	Input B

If not specified, CLK=1 is assumed.

MEAS FREQ [];

MEAS TIME [];

MEAS FRATIO [];

MEAS COUNT [];

These MEAS commands must be preceded by an appropriate SET FMU command. If any measurement except TR cannot be completed within 2.5 following a FMU MEAS command, a time-out occurs. To allow for measurements longer than 2.5 it is necessary to add a DLY statement that specifies enough time for the measurement to be complete **before** the MEAS command occurs. If a time-out occurs the error branch is taken, if one is included. With no error branch specified, an error message is presented on the display screen and the program pauses.

```
TEST FREQ <HI= LO=> [ ];
TEST TIME <HI= LO=> [ ];
TEST FRATIO <HI= LO=> [ ];
TEST COUNT <HI= LO=> [ ];
```

These commands tell the FMU to measure the specified quantity and then test it against the limits enclosed in inequality signs. If the test against limits fails, operation will branch to the command whose number is enclosed in square brackets. The measured value will be placed via the AR register in the appropriate register of those listed above.

Period is measured or tested in terms of frequency. There are no test language commands specifically addressed to period. FRATIO is Channel A divided by Channel B.

Some examples are included below which illustrate how the FMU might be programmed for the standard measurements:

Frequency Measurement:

```
100: SET FMU AT (3) TRIGA=0+;
133: MEAS FREQ [ ];
```

Fratio Measurement:

```
001: SET FMU AT (2:7) TRIGA=-2+ TRIGB=3-
      CLK=6;
007: MEAS FRATIO [  ];
```

Time Measurement:

```
010: SET FMU AT (1:1) TRIGA=5+ TRIGB=-5-
      PREC=8 INT;
050: MEAS TIME [  ];
```

Cumulative Count:

```
0017: SET FMU AT (7) TRIGA=0+ PREC=6 CLK=5;
0032: MEAS COUNT [  ];
```

Restrictions and requirements to be observed when writing SET FMU statements are shown below for each type of FMU measurement function. In the table, F=forbidden, R=required, and A=allowed.

	Frequency	Time	Ratio	Count
TRIGA	R only one required	R	R	R only one required
TRIGB	R	R	R	R
INT	F	R	F	F
PREC=1 thru				
PREC=5	A	A	A	F
PREC=6	F	F	F	R
PREC=7	A	A	A	F
PREC=8	A	A	A	F
CLK=1 thru				
CLK=3	A	A	F	A
CLK=4	F	F	F	F
CLK=5 (TRIGA)	F	F	R only one required	A
CLK=6 (TRIGB)	F	F	R	A
TR, ADIR, BDIR, and TIN are optional.				

CLEAR FMU;

This command causes the FMU to become disconnected from the DUT. The CLEAR ALL command will also produce the same result with the FMU.

3.2.15 Resistance Measurement Unit (RMU) [1792 D Option]

```
230: TEST RMU AT (8:10) <HI=10K LO=9.9K> [250];
```

The resistance measurement unit is connected to terminals 8 and 10. The conductance of the DUT circuit is compared to the specified resistance limits. If the test conditions are not met, the RMU register contains the actual resistance value. An optional branch address may be included in the command, as shown above, causing the program to branch when test conditions are not met. Note that if test conditions are met, the RMU register does not contain the measured value. The value is always stored in a register called RMU only with the MEAS command, as with the next example.

```
320: MEAS RMU AT (8:10);
```

The resistance measurement unit is connected to terminals 8 and 10. The conductance of the DUT circuit is measured and converted to a resistance value and stored in floating point form in the RMU register via the AR register.

```
330: CLEAR RMU;
```

This command causes the RMU to become disconnected from the source scanner. The CLEAR ALL command will also produce the same result.

3.2.16 Pulse Generator (PGEN)

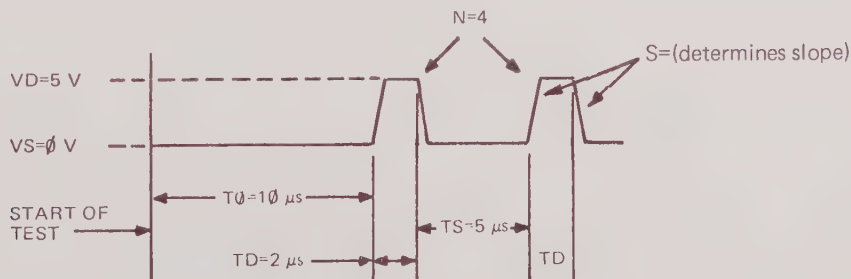
The pulse generator (PGEN) produces programmable pulse trains for use as DUT stimulus signals. If test system is equipped with two pulse generators (PGEN-1 and PGEN-2), the general form PGEN defaults to PGEN-1 in SET commands, and to both PGENs in CLEAR commands. Programming data are provided below.

SET PGEN-2 AT(5) N=4 T $\emptyset$ =10U TD=2U TS=5U VD=5 VS= $\emptyset$;

Required Parameters:

TD*	Pulse duration time, ranging from 40 nsec. to 21.474 sec.														
TS*	Pulse separation time, ranging from 40 nsec. to 21.474 sec.														
VD	Voltage level during pulse duration, ranging from 0 to ± 25 volts.														
VS	Voltage level during pulse separation, ranging from 0 to ± 25 volts.														
2	Selects PGEN-2. If not specified, PGEN-1 is assumed in SET commands.														
N	Transition count, ranging from 1 to 8 MEG**. A transition is a change from the VS to the VD state, or from the VD to the VS state. A single pulse is produced when N=2. Similarly, a low-to-high or high-to-low transition is produced when N=1; the output signal would remain in the resulting state until reset with a CLEAR PGEN or CLEAR ALL command, or until modified by a new SET PGEN command. If not specified, N=1 is assumed.														
T0*	Delay between the start of test and the first pulse transition, ranging from 40 nS to 21.474 sec. If not specified, T0 assumes the minimum value for the range that the TD and TS parameters are in.														
STROBE	The STROBE parameter initiates PGEN operations and timing based on the start of the next driver strobe except with 1799 systems. The keyword STROBE in 1799 systems enables a SET PGEN specification to be synchronized to an externally generated hardware-start signal (described later).														
S=	The slope of the PGEN waveform can be programmed. The slope in volts-per- μ sec can be included as listed below. If this parameter is not included, the slope (slew rate) is 500 V/ μ sec. <table><tr><td>S=1</td><td>500 V/μsec</td><td>S=4</td><td>50 V/μsec</td></tr><tr><td>S=2</td><td>200 V/μsec</td><td>S=5</td><td>8 V/μsec</td></tr><tr><td>S=3</td><td>100 V/μsec</td><td></td><td></td></tr></table>			S=1	500 V/ μ sec	S=4	50 V/ μ sec	S=2	200 V/ μ sec	S=5	8 V/ μ sec	S=3	100 V/ μ sec		
S=1	500 V/ μ sec	S=4	50 V/ μ sec												
S=2	200 V/ μ sec	S=5	8 V/ μ sec												
S=3	100 V/ μ sec														
CONT	A CONTInuous pulse train is generated until another SET PGEN or CLEAR PGEN command is encountered, or an external PGEN START signal (1799 only) is encountered.														

The example shown above results in the pulse train shown below:



*See PGEN range consideration (next page).

**The transition count is ignored when the CONTInuous mode is specified.

CLEAR PGEN;

This command disconnects both PGEN's from the DUT and ceases output of the associated signals. The CLEAR ALL command includes this same result. A single PGEN of test systems equipped with two pulse generators can be disconnected and de-activated without affecting the other one by explicit specification:

CLEAR PGEN-1; or CLEAR PGEN-2;

PGEN Range Consideration:

The T $\emptyset$, TD and TS parameters must all be specified within the same range. The range, however, is not explicitly stated within a command. The ranges are listed below. If T $\emptyset$ is not specified, the default T $\emptyset$ time is set by the system software to be the minimum value of the range that TD and TS are in.

<u>Range No.</u>	<u>Range</u>
1	40 nS — 335 mS
2	80 nS — 671 mS
3	160 nS — 1.342 Seconds
4	320 nS — 2.684 Seconds
5	640 nS — 5.368 Seconds
6	1.28 μ S — 10.7 Seconds
7	2.56 μ S — 21.474 Seconds

1799 Hardware Start:

If the optional keyword STROBE is included in a SET PGEN command, the statement is executed when a negative-going TTL pulse is received through shelf connector J30-1 (J30-2, Gnd) PGEN start.

3.2.17 Time Measurement Unit (TMU)

Three basic commands control TMU operation: SET, MEAS, and TEST. All measurements are made via special TMU scanner pins. Eight VF (start) and eight VS (stop) pins are provided. Terminal specifications included in SET TMU commands can only refer to these pins. TMU parameters are listed below. Each of the commands are used as described below.

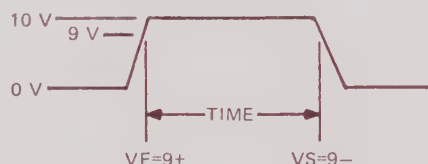
Parameter	Description
VF	First voltage threshold which starts the time measurement. A + or – sign must follow the VF specification to indicate a positive- or negative-going voltage threshold crossing. This is a required parameter for the SET TMU command.
VS	Second voltage threshold which stops the time measurement. A + or – sign must follow the VS specification to indicate a positive- or negative-going voltage threshold crossing. This is a required parameter for the SET TMU command.

SET TMU AT(2:5) VF=1.2+ VS=2.3–;

The SET TMU command programs operation parameters for the TEST TMU or MEAS TMU command(s) which follow. In the above example, START pin 2 is selected and assigned a 1.2 volt positive-going threshold. Similarly, STOP pin 5 is selected and assigned a 2.3 volt negative-going threshold. After executing this command, the TMU is "armed", and it is capable of executing the MEAS or TEST command. Other applications of the SET TMU command are shown in the following examples:

Pulse Width

SET TMU AT(3:3) VF=9+ VS=9–;



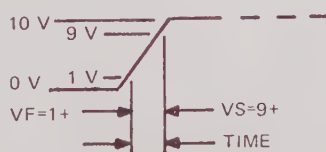
Time Between Pulses Or Events

SET TMU AT(3:3) VF=9– VS=9+;



Risetime

SET TMU AT(3:3) VF=1+ VS=9+; /RISE TIME OF 10 TO 90% OF 10 V PULSE/



80: TEST TMU <HI=3E-9> [220];

Each TEST TMU command must be preceded by a SET TMU command, and whatever other commands are required, to produce the event which will be timed. The measured value of time in seconds is stored in the TMU register (via the AR register), compared against the limit or limits given, and if the test conditions are not met, program operation will branch to the command number in the brackets. If the branch address is not included, the program will branch to a standard error handler. If for some reason either the start or stop signal, or both, are not generated in normal start-stop sequence within 2 sec., the branch to the address or error handler is taken. Either type of error will increment the FAIL register.

90: MEAS TMU [230];

Each MEAS TMU command must be preceded by a SET TMU command and other commands (normally one) to produce the event which will be timed. The measured value of time (in seconds) is placed in the TMU register (via the AR register). If for some reason either the start or stop signal, or both, are not generated in normal start-stop sequence, program operation branches to the command whose number is enclosed on the brackets. If no branch address is specified, the FAIL register is incremented and an error message is displayed. The TMU can be armed for measurement by external means (EXT ARM — digital shelf, BNC Connector AUX 3). EXT ARM **must** be held low (0 V - 0.4 V) during a SET TMU measurement. EXT ARM must then be held high (2.5 V - 5.0 V) for at least 100 μ s **prior** to the event to be measured and remain open at least 10 μ s longer than the event being measured.

3.2.18 Scanning System [1792 D Option]

The scanning system provides channel multiplexing between test equipment and shelf pins. For example, the command statement:

```
SET ACS AT (7) V=1.00 F=1K;
```

indirectly utilizes the scanning system to connect the AC source to pin 7 over source channel A (one of two source-connection channels). This same connection can be made by directly programming the scanning system with the following command statement:

```
SET SASCAN-1 AT (7);  
where: SASCAN selects source channel A, and  
      -1 identifies the test equipment (ACS)
```

Direct programming of the scanning system provides the flexibility for connecting auxiliary test equipment or for inter-connecting circuitry. The multiplexing of two source channels, two measurement channels and five power-supply channels to 36 shelf pins is directly programmable with SET and CLEAR commands.

NOTE

Programmed circuit shorts between source channels or among power supply channels are not executable.

Source Channel Multiplexing

Two source channels (A and B), each with three device connections (-1, -2, and -3) are directly programmable as follows:

Command Operand	Source Channel	Primary Device Assignment	Analog Scanner Box Connection Point
SASCAN-1	A	ACS	POS 15, J4
SASCAN-2	A	Spare (or PGEN-2)	POS 15, J3; or J4 on Shelf
SASCAN-3	A	RMU (I+)	POS 15, J1
SBSCAN-1	B	PGEN	POS 17, J1
SBSCAN-2	B	Spare (or ACS-2)	POS 17, J2; or J3 on Shelf
SBSCAN-3	B	RMU (I-)	POS 17, J3

A SET SASCAN or SET SBSCAN command without a test equipment device specification (-1, -2, or -3) interconnects the specified shelf pins via the source channel.

CAUTION

Connecting or disconnecting channels to circuits while current is applied can damage scanner contacts.

Only one device (–1, –2, or –3) can be connected to a single source channel at a particular time. Therefore, a SET source-channel command rescinds previous commands for that channel; including SET ACS (source channel A), SET PGEN (source channel B), and SET RMU (all source and measurement channels).

A source channel can be specifically disconnected with a CLEAR SASCAN command (disconnects source channel A) or a CLEAR SBSCAN command (disconnects source channel B). A CLEAR SCAN command clears all source, measurement, and power supply channels.

Measurement Channel Multiplexing

Two measurement channels (A and B), each with six device connections (–1 through –6), are directly programmable as follows:

SET command-operand AT (pin-list);

Command Operand	Measurement Channel	Primary Device Assignment	Analog Scanner Box Connection Point
MASCAN-1	A	DCV (+)	POS 4, J1-21 thru J1-24
MASCAN-2	A	ACV	Not Available
MASCAN-3	A	FMU (Channel A)	POS 6, J1
MASCAN-4	A	RMU (P+)	POS 6, J2
MASCAN-5	A	Spare	POS 6, J3
MASCAN-6	A	Spare	J5 on shelf
MBSCAN-1	B	DCV (–)	POS 4, J1-15 thru J1-18
MBSCAN-2	B	Spare	POS 7, J3
MBSCAN-3	B	FMU (Channel B)	POS 7, J1
MBSCAN-4	B	RMU (P–)	POS 7, J2
MBSCAN-5	B	Spare	POS 7, J4
MBSCAN-6	B	Spare	J6 on shelf

A SET MASCAN or SET MBSCAN command without a test equipment device specification (–1 through –6) interconnects the specified shelf pins via the measurement channel.

CAUTION

Connecting or disconnecting channels to circuits while current is applied can damage scanner contacts.

Only one device (–1 through –6) can be connected to a single measurement channel at a particular time. Therefore, a SET measurement-channel command rescinds previous commands for that channel, including the SET device commands for devices assigned to that channel.

A measurement channel can be specifically disconnected with a CLEAR MASCAN command (disconnects measurement channel A) or a CLEAR MBSCAN command (disconnects measurement channel B). A CLEAR SCAN command clears all source, measurement, and power supply channels.

Power Supply Channel Multiplexing

Five Kelvin-connecting power supply channels (A through E) are directly programmable using the SET PmSCAN command (where m represents the channel A through E as shown in the following charts (the primary use of each channel and its corresponding connection points are determined by the prevailing scanning-code-number of the system).

A power supply channel can be specifically disconnected with a CLEAR PmSCAN command where m is the desired channel designator (A through E). A CLEAR SCAN command clears all power supply channels as well as all source and measurement channels.

Chart – Programming Power Supply Channels

Command Operand	Channel	Primary Use By Scanning Code No.					
		1	2	3	4	5	6
PASCAN	A	Slot 1, HI					
PBSCAN	B	Slot 1, LO	Slot 2, HI				Slot 1, LO
PCSCAN	C	Slot 2, HI	Slot 3, HI	Slot 2, LO		Slot 4, HI	
PDSCAN	D	Slot 2, LO	Slot 4, HI			Slot 5, HI	Slot 4, LO
PESCAN	E	Slot 3, HI	Slot 5, HI		Slot 4, LO	Slot 7, HI	

Primary Use	PS Scanner Box Connection Points		
	POS	I	V
Slot 1, HI	10 & 11	J1-9	J1-6
Slot 1, LO	10 & 11	J1-21	J1-18
Slot 2, HI	10 & 11	J1-8	J1-5
Slot 2, LO	10 & 11	J1-20	J1-17
Slot 3, HI	10 & 11	J1-7	J1-4
Slot 4, HI	12 & 13	J1-9	J1-6
Slot 4, LO	12 & 13	J1-21	J1-18
Slot 5, HI	12 & 13	J1-8	J1-5
Slot 7, HI	14 & 15	J1-9	J1-6

3.2.19 External Test Equipment [1792 D Test System]

Five connectors on the rear of the analog scanner box and two on the analog shelf are provided for connecting external test equipment. One external source (XSOURCE-1) and three external measurement (XMEAS-i) devices can be connected as prescribed below. Both high and low sides of external device connections are scanned.

External Device		Analog Scanner Connection Location
XSOURCE-1	High	POS 17, J2 (Rear)
	Low	POS 15, J3 (Rear)
XMEAS-1	High	POS 6, J3 (Rear)
	Low	POS 7, J4 (Rear)
XMEAS-2	High	J5 (SCOPE 'A') Shelf
	Low	J6 (SCOPE 'B') Shelf
XMEAS-3	High	POS 7, J3 (Rear)
	Low	(Low terminal, which can be referenced to ground, is not scannable)

SET and CLEAR commands are provided for connecting and disconnecting the external devices. Programming examples are provided below:

SET XSOURCE-1 AT(2:5);

Connects XSOURCE-1 high output to terminal (shelf pin) 2 and low output to terminal 5.

SET XMEAS-1 AT(6:10);

Connects XMEAS-1 high input to terminal 6 and low input to terminal 10. Similar statements can be written for XMEAS-2. XMEAS-3 is used for single terminal measurement devices which are usually referenced to ground.

CLEAR XSOURCE-1;

Disconnects XSOURCE-1 analog scanner connections.

CLEAR XMEAS-2;

Disconnects XMEAS-2 analog scanner connections. Similar statements can be written for XMEAS-1 and XMEAS-3.

In addition, the CLEAR ALL command will disconnect all internal and external devices, including XSOURCE-1 and XMEAS devices.

3.2.2 0 Dc Voltage Measurement (DCVPRB) [Pulse-Catching Probe]

DC Voltage Probing

The pulse-catching probe of the Test System can be used to make nominal DC voltage measurements from -30 volts to +30 volts. A test language routine, such as the following, enables this feature:

```
200:  WRITE DEV=TEC '
      PLACE PROBE ON POINT U13.9,
      THEN PRESS CONTINUE ';
      PAUSE;
210:  MEAS DCVPRB;
```

The above routine displays an operator instruction on the screen of the data terminal and then waits for operator compliance. When the operator presses the CONTINUE pushbutton of the control panel, the DC voltage at the probe is measured and stored in a register (DCVPRB) for subsequent reference. If it is desired to display the measured nominal value to two decimal places, the following commands can be appended to the above routine:

```
      FORMAT 6,2;
      WRITE DEV=TEC'
      RESULT='DCVPRB' VOLTS';
```

Omitting the FORMAT command causes the value in the DCVPRB register to be displayed in accordance with the most recently established format commanded by a previous FORMAT statement.

Instead of making a simple measurement, a conditional program branch can be performed based on the DC voltage at the probe by using a command statement such as the following in place of statement 210 in the above routine:

```
210:  TEST DCVPRB <HI=5 L0=3> [330];
```

This TEST DCVPRB command stores the value of the DC voltage at the probe in the DCVPRB register and tests the value against the specified limits (<HI=5 L0=3>). If the value is between 3 and 5, the program proceeds to the next command statement. If the value is not within the limits, the program branches to statement 330 as specified by the error branch ([330]).

3.3 ARITHMETIC COMMANDS [All Test Systems]

3.3.1 General

Arithmetic commands consist of two fields;

1. **Function Field.** This field consists of a word such as ADD, SQRT, MULT, LOAD, etc.
2. **Operand Field.** This field contains a register name, a numerical value (literal constant), or nothing.

Associated with the arithmetic commands is a group of storage registers:

1. **Arithmetic Register (AR).** This is the working register. The results of all arithmetic operations appear in this register.
2. **REG-1 through REG-100.** These registers provide general data storage for arithmetic operations and are available for use in test programs.
3. **Measured Value Registers.** These registers are dedicated for storage of measured data values obtained through RMU, DCV, DCI, etc. operations. Refer to Table 2-6 for a complete list of registers and their associated reserved words (names) by which they are addressed.

The paragraphs which follow include examples of arithmetic commands and a description of each.

3.3.2 Load (LOAD or LD)

```
30: LD REG-3;  
110: LD ACV;  
210: LD 1.35;  
225: LD 1.45E-3;
```

The first two examples cause the contents of the specified register to be loaded into the AR. The previous contents of the AR are lost. The latter two examples indicate the method of loading literal number values into the AR for use as constants.

3.3.3 Store (STORE or STO)

```
270: STO REG-4;      STO TIME;
```

The contents of the AR are stored in the designated register. The previous contents of the storage register are lost; AR contents are not changed.

3.3.4 Add (ADD)

```
215: ADD REG-13;  
216: ADD DCV;  
220: ADD 2.71828
```

The contents of the specified register or the value of the literal constant is added to the contents of the AR; the result (sum) remains in the AR.

3.3.5 Subtract (SUB)

```
222: SUB REG-12;  
231: SUB 32;
```

The contents of the specified register or the literal constant value is subtracted from the AR; the result is placed in the AR.

3.3.6 Multiply (MPY or MULT)

20: MPY 4.2E2;
30: MULT REG-1;
40: MPY ACV;

The value in the AR is multiplied by the specified constant or contents of the specified register. The result is placed in the AR.

3.3.7 Divide (DIV)

70: DIV 2.931;
120: DIV REG-10;

The contents of the AR are divided by the literal constant or contents of the specified register. The quotient is placed in the AR.

3.3.8 Square Root (SQRT)

250: SQRT;

The square root of the contents of the AR is computed and the result is placed in the AR.

3.3.9 Square (SQUARE)

270: SQUARE;

The square of the contents of the AR is computed and the result is placed in the AR.

3.3.10 Absolute (ABSOLUTE)

Causes the signed contents of AR to be replaced with the absolute (unsigned) value of AR.

3.3.11 Arithmetic Test (ATEST)

ATEST REG-4 <HI=2.3 LO=1.9> [70];
ATEST REG-4 <LIM=2.1 + 10% -10%> [90];

These command examples (arithmetic test) compare the value in register 4 against the limits given. As stated previously, the limits specified can be expressed in alternate ways (refer to paragraph 2.3). If the test conditions are not met, the program operation branches to the command whose number appears in the brackets if specified; otherwise, system message is displayed. A failure to meet the test conditions in an ATEST command increments the FAIL register.

3.3.12 Comparison (COMP)

220: COMP ACV <HI=2.3 LO=1.9> [70];
270: COMP TMU <HI=REG-3> [337];

This type of command is similar to the ATEST command with one exception: it does not increment the FAIL register if the test conditions are not met. The COMP command will, however, branch to the command whose number is in the brackets. (error branch specification is required in COMP command statements). The COMP command has application in conditional branching when it is not desired to flag occurrence of an error.

3.3.13 Trigonometric Operations

All commands for trig. operations operate on the contents of the AR. The AR must be loaded with data in radians prior to executing the SIN and COS commands. ATAN results are in radians. Commands are listed below:

Command	Description
SIN;	SIN(AR)→(AR)
COS;	COS(AR)→(AR)
ATAN;	TAN ⁻¹ (AR)→(AR)

NOTE

The following conversion factors may be used with the Multiply command to convert between degrees and radians.

Radians = 0.01745329 x Degrees
Degrees = 57.2957795 x Radians

3.3.14 Logarithmic Operations

All commands for natural log (logs) operations operate on the contents of AR. Hence, the AR must be loaded with data prior to executing the log command. Two log commands are provided, as listed below:

Command	Description
LOG;	Log _e (AR)→(AR) (see note below)
EXP;	e(AR)→(AR)

NOTE

Illegal arguments for the LOG command (AR contents are zero or negative) result in the AR being unchanged.

3.4 UTILITY COMMANDS [All Test Systems]

3.4.1 General

Utility commands perform the miscellaneous functions not previously categorized but which enhance the fluency of test programs. The following paragraphs and command-statement examples describe the various utility commands available.

3.4.2 No Operation

NOP;

NOP is a no operation command; no test system execution is performed by this command.

3.4.3 Block

BLOCK;

The BLOCK command marks the end of a core boundary. During translation (TRANS), an automatic blocking feature of the system inserts BLOCK commands automatically so as to not exceed the 4K boundaries of core. Each automatic insertion of a BLOCK command causes a printout of a warning message that includes the label-referenced location of the command statement immediately preceding the inserted BLOCK command. To avoid program transfers across core boundaries resulting from program branching, DO loops, etc. the user can override the automatic blocking feature by appropriately locating BLOCK commands manually.

Under exceptional circumstances during translation, an error code 16 (block too large) may occur to indicate that manual insertion of a BLOCK command is required.

Do not insert a BLOCK command within a critical timing sequence.

3.4.4 End

END;

The END command signals the end of the test program. It is required in all test programs. The command terminates program execution, clears all test system equipment, including scanners, stimulus, measuring devices, etc.

3.4.5 Branch

BRANCH 123;

The BRANCH command transfers program control to the test statement specified (label 123 in the above example).

BRANCH 'NAME';

All branching to filenames requires that the corresponding binary file (translated program with a .TA, .TD, or .TL extension) exist on the second working disk specified for DIAGNOSIS Mode (see GR1792A thru D Operation instructions). Executing the BRANCH 'NAME' command clears any DO or CALL operations but does not alter the contents of registers.

BRANCH commands can be used as optional error branch specifications for SET, MEAS, TEST, ATEST, COMP, CCOMP, MCOMP, and CALL commands when enclosed by brackets, e.g., [BRANCH 250], which can be conveniently abbreviated to [250].

When branching to LGC command statements, the translation (or physical) order of the statements determines the logic states (or levels) of the applicable driver/sensor pins. Consider the following series of commands.

	Input Pins			
	1	2	3	4
10: \$IL(#);	0	0	0	0
20: \$IH(1,3);	1	0	1	0
30: \$IH(2,4);	1	1	1	1
40: \$IL(1,3);	0	1	0	1
BRANCH 20;				

When the BRANCH 20 command is encountered, the logic states of input pins 1 through 4 are 0101 (the result of statement 40). But after execution of the BRANCH 20 command, the logic states of the input pins are 1010 (the states associated with statement 20), not 1111 as might be expected.

3.4.6 Call

```
CALL 125;
```

The basic CALL command transfers program control to a sequence of instructions beginning with the statement label specified by the CALL command (statement 125 in the above example). Program control returns to the statement following the CALL command when a RETURN command is encountered in the sequence.

Error-branching can be specified by a CALL statement as shown in the following example.

```
CALL 126 [10];
```

In this example, an ERRET executed by the subroutine at 126 transfers program control to an instruction with statement label 10 (error-branching specification).

Basic CALL commands can be used as optional error branch specifications for SET, MEAS, TEST, ATEST, COMP, CCOMP, MCOMP, and other CALL commands when enclosed by brackets ([]), for example.

```
ATEST REG-4 <HI=2.3 LO=1.9> [CALL 300];
```

In the above statement, the error branch is a CALL to the subroutine at 300. If the value in REG-4 is not between 1.9 and 2.3 (inclusive), the subroutine at 300 is performed. Upon completion of the subroutine (RETURN command encountered), program control is transferred to the statement following the ATEST command.

CALL commands may include active pin lists as follows:

```
CALL 250 (5, 7, 9);
```

This statement specifies that pins 5, 7, and 9 are the only pins freely active during the course of the subroutine at 250; these are the only pins whose states can be changed by the instructions of the subroutine. The states of all other assigned pins remain as they were when the CALL command was encountered.

NOTE

A CALL command without an active pin list (the basic CALL command) implies an active pin list with all assigned pins listed.

As with the **BRANCH** command, the logic states (or levels) of applicable driver/sensor pins are determined by the translation (or physical) order of the **LGC** command statements of the program. The following series of commands with an associated input-pin state table for each pass through the commands illustrates the effects of the translation order and also the application of a **CALL** command with an active pin list.

		Input Pins					
		Pass 1			Pass 2		
		1	2	3	1	2	3
10:	\$ IL(#)	0	0	0			
20:	\$ IH(3);	0	0	1	1	0	1
30:	\$ IH(2) IL(3);	0	1	0	1	0	0
40:	\$ IH(3);	0	1	1	1	0	1
	RETURN;						
50:	\$ IH(1) IL(2,3);	1	0	0			
	CALL 20(3);						
60:	\$ IH(2) IL(1);				0	1	0

In the above example, statements 10 through 50 control the logic states of pins 1 through 3 as shown by the state table for pass 1 (the **RETURN** command is ignored during the first pass because no **CALL** command is active). Execution of statement 50 establishes 100 as the logic states of pins 1 through 3 respectively. The **CALL** command calls for execution of the subroutine consisting of statements 20 through 40 and ending with the **RETURN** statement. The **CALL** command also includes a pin list that freezes the current logic states of all pins except pin 3 (the only active pin listed). Therefore, the current logic states of pins 1 and 2 as established by statement 50 are frozen at 1 and 0 respectively while the **CALL** command is active. This is reflected in the state table for pass 2. Pin 3, however, is freely active and is allowed to assume the logic state previously established for statements 20 through 40 during translation. This also is reflected in the state table for pass 2. When the subroutine is completed, the **RETURN** command de-activates the **CALL** condition, the logic states of all pins revert to the states they were in immediately prior to the **CALL** command (as established by statement 50 in the above example), and program control is transferred to the statement immediately following the **CALL** command (statement 60). Since statement 60 does not specify a logic state for pin 3, the logic state established for statement 50 during translation prevails (not the logic state established during execution of the subroutine).

3.4.7 Return

RETURN;

This command causes program control to be transferred back to the calling program segment, to the command following the **CALL** command. If no **CALL** command had been previously issued causing execution of the present program segment, including the **RETURN** command, the **RETURN** command is ignored.

RETURN commands, when enclosed by brackets ([RETURN]), can be used as optional error branch specifications for **SET**, **MEAS**, **TEST**, **ATEST**, **COMP**, **CCOMP**, **MCOMP**, and nested **CALL** commands in called program segments to immediately return program control to the calling program upon error sensing.

3.4.7A Error Return

ERRET;

This command, in a called program segment first performs a **RETURN** command, then transfers program control as specified by the error branch specification of the associated **CALL** command in the calling program. If no **CALL** command is active, the **ERRET** command is ignored.

ERRET commands, when enclosed by brackets ([ERRET]), can be used as optional error branch specifications for **SET**, **MEAS**, **TEST**, **ATEST**, **COMP**, **CCOMP**, **MCOMP**, and nested **CALL** commands in called program segments to transfer program control as specified by the calling program upon error sensing. For convenience, [ERRET] can be abbreviated to [].

3.4.8 DO Loops

```
DO 5;          DO REG-1;    /THE VALUE OF REG-1 UPON ENTRY/  
...;          ...;  
...;          or           ...;  
DONE;         DONE;
```

DO loops are bounded by the DO and DONE commands. In the example shown above, the commands located between the DO and DONE commands are executed five times. An operand which specifies the number of times that the loop is to be executed must follow the DO command as shown. After completing the required number of loops, the test program executes the command following the DONE command. Any fractional part in the number of loops specified are truncated. DO loops can be nested to a level of nine (maximum).

NOTE

DO loops cannot be used within CAPS recording areas if the loops contain TEST LGC commands. SET LGC (\$) commands are, however, permitted. (See REPEAT/LOOP MACROS).

ESCAPE 20;

The ESCAPE command is used for program branching out of a DO loop prior to completing the programmed number of loop executions. The program's active DO loop index is cleared so that it appears that the number of DO loop executions have been completed. An address label must be included as a parameter. For example:

```
DO 8;  
MCOMP LGC OL(3) [10];  
ESCAPE 20;  
10:DONE;  
20:next command
```

If the MCOMP conditions are not met, the program will branch to 10, the DONE command, causing the DO loop to be executed again until eight loop executions have been completed. After the final loop, the loop index is exhausted, and the command following the DONE command is executed. However, if the MCOMP conditions are met, the next command (ESCAPE) is executed, causing program branching to the command at label 20, and the loop index is reset, completing the DO loop execution.

3.4.9 Programmed Delays

DELAY 35M;

The DELAY command inserts a no-operation time delay, specified by the operand. In the example shown, a minimum delay of 35 ms is inserted before executing the command following the DELAY command. The range, in which time delays can be specified, is from 1 ms (1M) to 511 sec (511) in increments of 1 ms.

3.4.10 Pause

PAUSE 10; or PAUSE REG-5;

This instruction causes a temporary stop in the test program execution. The operator must press the CONTINUE button on the control panel in order to resume program execution. An optional numeric parameter may be inserted, such as the number 10 in the above example, which is displayed on the data terminal during the pause. Similarly, the numeric contents of a general register could be displayed by including the register's name. The use of a PAUSE command allows suspension of program execution while some external operation is performed by the operator.

3.4.11 Clear All Test Equipment

CLEAR ALL;

The CLEAR ALL command causes all test equipment to be cleared in exactly the same manner as the following sequence of individual CLEAR commands.

CLEAR LGC;
CLEAR RLY;
CLEAR PGEN;
CLEAR ACS;
CLEAR FMU;
CLEAR RMU;
CLEAR ACV; CLEAR DCV; CLEAR DCI;
CLEAR PS;
CLEAR SCAN;

This CLEAR ALL sequence is also performed at system initialization; which occurs at program start, test program failure (if CONTINUE ON ERROR is off) and program END. However, when caused by test program failure, all driver/sensors are changed to sensors but the logic level assignments are left intact.

3.4.12 Character Comparison (CCOMP)

280: CCOMP 'CHARACTER STRING' [350];

This command compares the contents of the CHAR register (up to 20 characters) with the character string in the parameter field bounded by single quotes. If the character data do not compare, the program branches to the command at the branch address specified and a CHAR COMPARE ERROR message is displayed. The branch address is a required parameter. This command does not increment the FAIL register.

3.4.13 Input/Output Commands

Writing Commands

OPEN WRITE 'LOGFIL.LG' 2;

This OPEN WRITE command creates and defines a logging file (LOGFIL.LG) with a guaranteed minimum length of 2 disk blocks (1000₈ words) on an OS/8 structured device specified to the Monitor. The file name of the logging file is user assigned. When an extension for the logging file is not specified by the user, the system assigns an extension of .LG. The file length argument, which is optional, is a floating-point number that specifies the minimum quantity of disk blocks required. When not given, the system will default to the largest empty area working device Ø.

CLOSE;

The CLOSE command closes the currently open logging file and places it in the directory. This command is used following the last WRITE DEV=FILE statement that is directed toward the particular logging file.

WRITE DEV=TEC 'TEST FAILED' REG-1 'MORE TEXT';

This command allows outputting text character strings and register values to the named device for display. For the above example, the Data Terminal will display TEST FAILED, followed by the floating-point value contained in REG-1, followed by MORE TEXT. Note that the character strings must be bounded with single quotes; registers whose values are to be displayed are not enclosed in quotes. Anything is valid between single quotes except another single quote or the % character, unless a special character is specified as described in paragraph 3.4.13.2. The writing device can be specified by either its symbolic name or numeric designator as follows:

DEV=PRINT or 0 (default)	Write on Teletype or line printer, as selected by listing device option LI= and enabled by ON status of print option D (DIAGNOSIS Mode options) If not enabled, TEC is default device
DEV=TEC or 1	Data terminal
DEV=TTY or 2	Teletype
DEV=PTP or 3	High-speed paper-tape punch
DEV=LPT or 4	Line printer
DEV=FILE or 5	Logging file, as created by an OPEN WRITE command.

FORMAT 6, 2;

The format command establishes a write format to a field width (ie — the total number of digits to be displayed, including sign and decimal point;) specified by the first operand and the number of digits after the decimal point specified by the second operand. The operands are separated by a comma, as shown. The maximum field width is 16. The above example would cause the value — .1612345 E+2 to be displayed in the following format:

—16.12

To re-establish the E format, use the following command:

FORMAT EXP;

Special Data Terminal Characters

Percentage characters allow special data terminal commands (that are normally a function of GR System software) to be used by the test programmer. A character can be used to position text anywhere on the screen, make text blink at a 2 Hz rate, activate a data terminal keyboard bell generator to signal an operator or perform a series of other special functions. Percentage characters must be combined with WRITE statements in the test program. Table 3-1 lists the percentage character functions available. Some of the more useful functions are described in the following paragraphs.

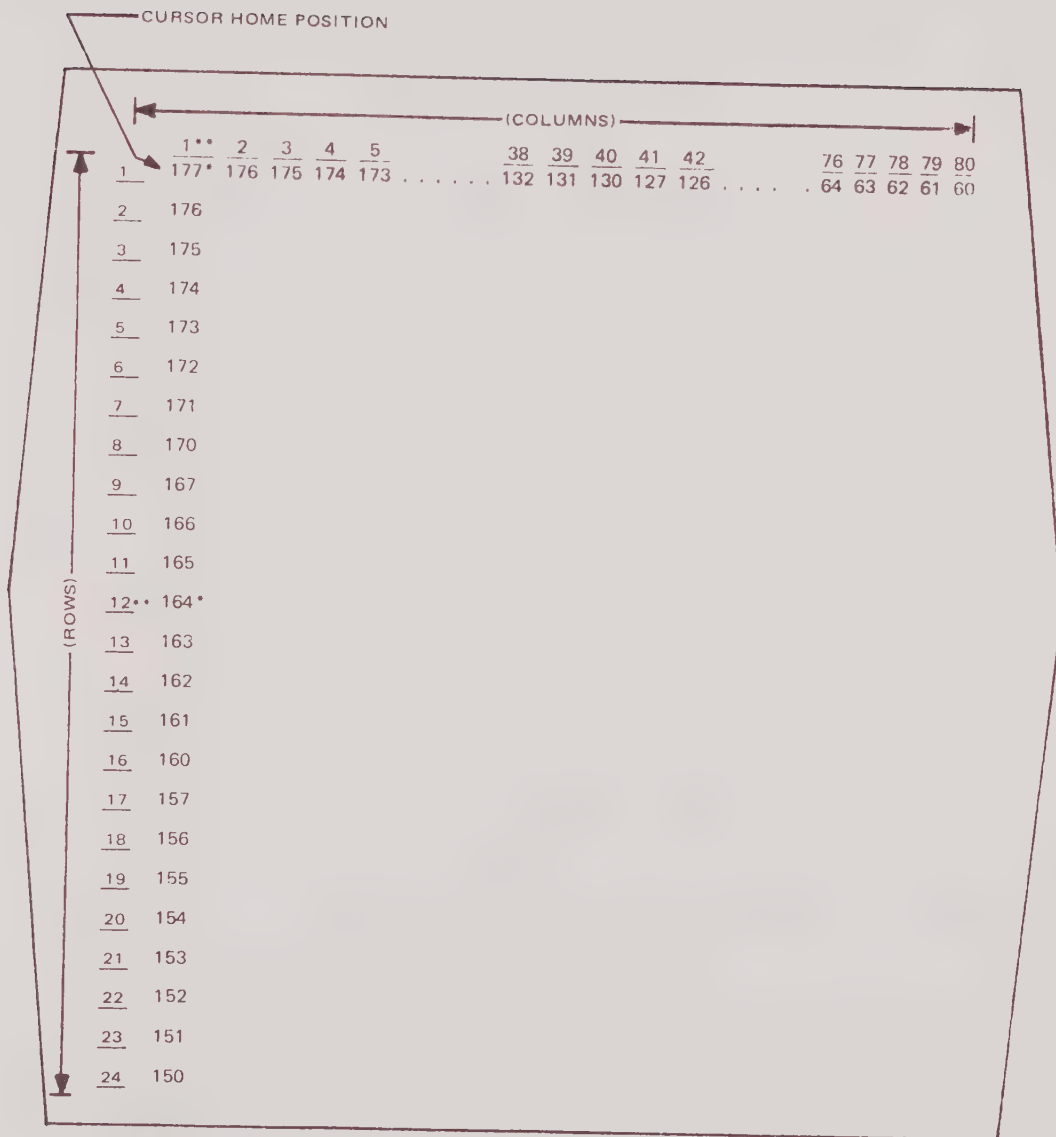
For example, the text string:

'%015'

causes the test program to send the 3-digit octal code 015 to the named I/O DEvice. Three octal digits *must* be included for each special character. Generally, these octal digits are ASCII codes (refer to the instruction manual for the write DEvice for a listing of I/O codes). A string of more than one special character can be included in the WRITE statement by including the % character, immediately followed by the correct octal code for the special character, for each special character in the string. Para. 3.4.13.2 describes special characters used with the data terminal.

Positioning Text on Data Terminal

Up to a maximum of 24 text lines (each of which may contain a maximum of 80 characters) can be displayed on the screen at any one time. Thus, a maximum of 24 horizontal rows (text lines) and 80 vertical character columns make up the total message area available for use (see Figure 3-1). Note that the HOME position of the cursor is the upper left corner of the screen (column 1, row 1). To position text anywhere on the screen, it will be necessary to determine the octal number that designates the desired column position, and the octal number that designates the desired row position. Both of these numbers are combined with the percentage character to position the cursor anywhere on the screen. Text positions are determined by the position of the cursor.



*Octal Numbers.

**Decimal Numbers.

Figure 3-1. Data Terminal Screen Message Areas.

Table 3-1
PERCENTAGE CHARACTER FUNCTIONS

Octal Code	Function
007	Sounds bell tone in data terminal keyboard.
015	Carriage Return (CR) . Moves cursor to left end of text line. ⁽¹⁾⁽²⁾
040	Space (SP) . Inserts a blank character. ⁽¹⁾
045	Percentage character (%)
047	Apostrophe (')
143	Erase to End Of Line (EEOL) from cursor position.
144	Insert Character (IC) . Opens position over cursor.
145	Insert Line (IL) . Opens line above cursor.
146	Clear Screen (CS) . Clears entire screen. Will not clear protected areas.
147	Moves cursor one position to the right (→).
150	Moves cursor one position down (↓).
151	Homes cursor. Moves cursor to top left position of screen (column 1, row 1).
154	Positions cursor at column and row designated.
163	Erase to End Of Page (EEOP) from cursor position.
164	Delete Character (DC) over cursor and close up text.
165	Delete Line (DL) over cursor and close up text.
166	Clear Memory (CM) . Clears entire screen including protected areas.
167	Moves cursor one position to the left (←).
170	Moves cursor one position up (↑).
171	Start format protection with the next character. ⁽¹⁾
172	End format protection at the last character. ⁽¹⁾
173	Start blinking at the next character. ⁽¹⁾
174	Stop blinking at the last character. ⁽¹⁾

(1) Appears as a blank space in displayed text lines.

(2) A carriage return can also be represented in text by an up arrow (↑), which is entered using the SHIFT and N keys.

NOTE

Text displayed beyond a 72 character limit cannot be printed by teletype.

The three digit octal number that represents the cursor upper left (or HOME) position is 177 (column 1, row-1). To find any other character column position across the screen, refer to Figure 3-1.

For example, if it were desired to start a text string 8 character columns to the right, and 12 line rows down (from the HOME position), the two three digit octal positioning numbers required would be: 170 (Octal column number) and 164 (Octal Row number). These numbers are preceded by the octal code 154 which is a position command.

Once the column and row numbers have been determined, the program statement can be prepared. In the following example,

```
WRITE DEV=TEC '%154 %170 %164 MESSAGE TEXT';
```

the percentage positioning numbers have been combined with a WRITE statement. The first % character (154) positions the cursor at the column (170) and row (164) designated. The message that follows will be displayed at the coordinates indicated.

NOTE

1. Omission of the percentage positioning numbers from a WRITE command to the TEC device causes the message to begin at the existing location of the cursor.
2. Screen positioning data (% numbers) can cause random characters when written to devices other than TEC.

Blinking Text on Screen

If it were desired to make the message IC 15 HAS FAILED blink at a 2 Hz rate, the following format could be used:

```
WRITE '%173 IC 15 HAS FAILED %174';
```

Ringling Data Terminal Keyboard Bell

The following message format will activate the data-terminal keyboard bell generator:

```
WRITE 'IC 15 HAS FAILED %007';
```

If it were desired to have the message blink and also ring the bell, the following format could be used:

```
WRITE '%173 IC 15 HAS FAILED %174 %007';
```

Rules For % Characters

1. Percentage characters must be combined with WRITE statements.
2. When positioning the cursor, the column number must occur first (ahead of the row number).
3. All numbers used must be octal. Octal numbers are automatically converted to a machine readable format during translation. Three octal digits must be included for each character (007, *not* 7).

Reading Commands

```
READ DEV=TEC R;
```

This command causes the transfer of a real numeric value from the device named into the AR; number reading continues until a carriage return is read or an invalid character is discovered. The R may be replaced by a C, causing a character string to be read into the character register (CHAR); character reading continues until a carriage return is read or until 20 characters have been read in (the data terminal will erase characters in excess of 20). Operands include the device designator (unless the default device is desired) and either the R (default) or C parameter. Parameters are defined below:

DEV=FIRST or 0 (default)	Read from any device having data ready
DEV=TEC or 1	Data terminal
DEV=TTY or 2	Teletype
DEV=PTR or 3	High-speed paper-tape reader
R (or default)	real number
C	character

When reading from Data Terminal, input data must begin at the cursor HOME position and conditioned for the READ operation by actuation of the XMIT key.

If a typing error occurs when READING from the teletype keyboard, press the RUB OUT key to delete the entire command line. The correct command line can then be typed. The line is terminated by the carriage-return key, by 20 characters if a character read, or by an illegal character if a floating-point-number read.

3.4.14 Special Program Execution (RESERVE, EXEC)

Two special test-language commands (RESERVE and EXECute) enable the test-language program to summon user-generated PAL8 language routines. These PAL8 routines, assembled as either subroutines or overlays, are invoked by EXECute statements. A RESERVE statement, required in test programs that invoke PAL8 routines assembled as subroutines for use on test systems with the minimal complement of core (16K words in 1792A/B or 20K words in 1792C/D), specifies the amount of core area to be preserved for the subroutines.

CAUTION

Much care must be used when including these commands in the test program, since they transfer control out of the test system; an error in the assembly language portion of the program or routine being executed could destroy all software systems, resulting in a considerable loss of time. Therefore, use of these commands should be restricted to experienced assembly language programmers who thoroughly understand the structure of the software systems in use.

NOTE

Guidelines and techniques, available to the assembly-language programmer for the writing of overlays and subroutines, are highlighted in Appendix B.

```
RESERVE 2100;
```

The RESERVE statement consists of the keyword RESERVE and a one-to-four digit octal value that specifies the number of word locations to be preserved in each core memory field available for storing user-generated PAL8 language subroutines (not available are fields 0 through 2 in 1792A/B systems and fields 0 through 3 in 1792-C/D systems). A maximum of 3400 word locations can be RESERVED beginning with word location 0. In the above example, 2100 word locations are specified. This specification is rounded upward to include complete 400-word system blocks so 2400 word locations from 0 to 2377 in each available core memory field are thereby preserved.

NOTE

RESERVE, if used, *must* be the first statement in the test program. If the test program includes a branch to another test program, that program *must also* contain the *identical* RESERVE command as its first statement. The RESERVE command *must not* include a label.

```
EXEC 50312 [440] (OCTAL 1777, 'A3Q5-B', -14.80);
```

As shown above, the EXECute statement for invoking subroutines consists of the keyword EXEC, a five-digit starting address of the subroutine in terms of memory field(5) and desired entry location (0312), and an optional data list enclosed between parentheses. An optional error branch ([440]) is specified in the example. Positionally, the error-branch specification must follow the operand of the EXECute command and precede the data list, if any. Note, however, [CALL XX]-type error branches are not permitted in EXEC statements.

The EXECute statement for invoking overlays, as shown below, also consists of the key word EXEC and an optional data list enclosed between parentheses. The difference is that a one-to-six character filename of the overlay (TOUT32 in this case) is specified between apostrophes. If desired, although not required, the SYStem device and filename extension may be included in the specification, e.g. 'SYS:TOUT32.SV'.

```
EXEC 'TOUT32' (OCTAL 20, 'ALL 1%247S', REG-3);
```

Optional data lists, which are enclosed between parentheses, are employed only to pass data items required by the particular subroutines or overlays. Within the list, data items are separated by commas and can be any required combination of octal values, alphanumeric text, floating point numbers, and P NAMES. If symbolic names (P NAMES) are used within EXEC commands, symbolic names cannot be register names (para. 2.7) or the word OCTAL.

An octal-value data item requires the keyword OCTAL followed by a blank (space character) followed by up to four octal digits (corresponds to 12 binary bits). Leading zeros may be omitted, e.g. OCTAL 20.

An alphanumeric-text data item must be enclosed between apostrophes and can be any alphanumeric text string of variable length. Rules governing the text string are the same as those that apply to the test-language WRITE command, including the percentage-character convention for special characters and functions. This convention is employed in the preceding command example to represent the apostrophe character in ALL 1'S.

A floating-point-value data item can be represented as a decimal number (–14.80), in exponential form (–.148E2), or by the symbolic name of the register containing the value (REG-3).

3.4.15 Symbolic Terminal Designations (ASSIGN PNAME)

Symbolic addressing can be used for terminal designations. To assign a symbolic name, the ASSIGN PNAME statement is used, referencing the symbolic name and the actual terminal designation. For example:

```
ASSIGN PNAME AB1=123,AB2 =124,X567=64;
```

assigns AB1 as the symbolic name for test system pin 123, AB2 for test system pin 124, and X567 for test system pin 64. The following rules must be observed when using symbolic names :

1. The ASSIGN PNAME command, which defines the symbolic name, must precede the first command referencing the name.
2. A pin cannot be defined by using a symbolic name for the actual terminal designation (ASSIGN PNAME AB1=XYZ12 is an invalid command).
3. The test system will accept up to six characters in a symbolic name.
4. The symbolic name *must* begin with an alpha (any letter) character; the remainder of the symbolic name may consist of any combination of letters or numbers, as desired. *Do not* use symbols in the name.
5. The ASSIGN PNAME command contains no label.
6. Up to 511 symbolic names can be used for any terminal designation, including all driver/sensor (logic), source/measurement multiplex scanner (analog, TMU scanner, relay drivers, and power supply scanner pins.*
7. Symbolic (PNAME) designations are not permitted in SET PULSE statements.
8. A PNAME is passed as a floating-point value in EXEC data-lists (para. 3.4.14).**

After symbolic designations have been assigned, the pname and integer-form of terminal designation may be used interchangeably in subsequent program statements except as indicated by rule 7 above.

*The translator responds with a warning message if a PNAME has been assigned that conflicts with a scanner keyword.

**The translator responds with a warning message if a PNAME has been assigned in an EXEC data-list that conflicts with any EXEC data-list keyword.

3.4.16 User Option Switches (USOPT1, USOPT2)

Two hardware control panel push-buttons, USOPT1 and USOPT2 provide flexibility for operator-influenced test program branch control. When the User Option 1 Switch is pressed the USOPT1 software register is set to "1". Further pushing of the button results in the complement of the register. USOPT2 functions the same way. These switches are useful for manual test program direction, since a program branch can occur depending on the state of a register by using a command similar to the following:

```
COMP USOPT1 <HI=.5>[100];
```

In this example, if the control panel User Option 1 Switch has been pressed (associated lamp is on), then register USOPT1=1. For the example above, the test against limits fails and the branch is taken. Refer also to the more complex example in "Register Usage" in Appendix C.

Two other associated commands can be used to control user option switch activities within the test program:

```
SET USOPT2;
```

The control panel User Option 2 lamp is turned on, and the register USOPT2 is set to 1.

```
CLEAR USOPT1;
```

The control panel User Option 1 lamp is turned off, and the register USOPT1 is reset to "0".

3.5 MACRO PROGRAMMING AID [All Test Systems]

A repertoire of macro commands is available to ease the burden of repetitively coding sequences of test-language command statements. The macro processor option of the TRANSLate Mode (or PREPare Mode in CAPS-equipped systems) produces a source-language test program with the written macro commands replaced by corresponding sequences of test-language command statements. Macro commands can reiterate bounded sequences of test-language commands (REPEAT/LOOP macros), can create sequences of TEST LGC or SET LGC commands with varying input specifications (System Macro Commands), and can insert user-defined sequences of test-language commands containing variables (User-Styled Macros).

3.5.1 REPEAT/LOOP Macros

The REPEAT/LOOP macros are similar to the test-language DO and DONE commands, but with certain advantages. These advantages are a result of the fact that macros are transparent at time of test program execution; thus minimizing execution time. This transparency of the REPEAT/LOOP macros also offers the programmer a means for repeating testing segments within CAPS RECORDing areas where TEST LGC statements within DO and DONE commands are not permitted. The following program segment illustrates the use of REPEAT/LOOP macros in a written program.

```

/START OF OUTER REPEAT/
. REPEAT 2;

120:    $IL(ENABLE)      /INDENTED FOR READABILITY/;

        /START OF INNER REPEAT/
15:    . REPEAT 3;
        $IH(COUNT);
        $IL(COUNT);
        . LOOP;
        /END OF INNER REPEAT/

        MCOMP LGC OH(AH13) [15];
. LOOP;
/END OF OUTER REPEAT/
```

The REPEAT and LOOP keywords are each prefixed with a period (.) to identify them as macro commands to the macro processor option of the TRANSLate Mode. The integer contained in the REPEAT command statement specifies the number of times the sequence bounded by the REPEAT and its associated LOOP macro command is to be reiterated. The above example contains two bounded sequences; one inside the other. The outer sequence is to be repeated twice. The inner sequence is to be reiterated three times within each repetition of the outer sequence. Macro processing of this written program segment results in the source program segment shown on the next page. As shown, the result has been enlarged as directed by the REPEAT/LOOP macros and now consists entirely of test-language statements (the macros have been removed).

In this example, an MCOMP LGC command (with an error-branching label used in an inverse fashion) simulates the so-called escape from the loop. This command also exemplifies the automatic renumbering feature of the macro processor. This feature renumbers (in ascending sequence) all labeled statements of the written program. When labeled statements are repeated as a result of the REPEAT/LOOP macros, each statement repetition is labeled with the next ascending number. Branching and error-branching labels are renumbered accordingly to maintain program integrity.

/START OF OUTER REPEAT/

5: \$IL(ENABLE) /INDENTED FOR READABILITY/;

 /START OF INNER REPEAT/

10:

 \$IH(COUNT);
 \$IL(COUNT);

 \$IH(COUNT);
 \$IL(COUNT);

 \$IH(COUNT);
 \$IL(COUNT);

 /END OF INNER REPEAT/

 MCOMP LGC OH(AH13) [10];

15: \$IL(ENABLE) /INDENTED FOR READABILITY/;

 /START OF INNER REPEAT/

20:

 \$IH(COUNT);
 \$IL(COUNT);

 \$IH(COUNT);
 \$IL(COUNT);

 \$IH(COUNT);
 \$IL(COUNT);

 /END OF INNER REPEAT/

 MCOMP LGC OH(AH13) [20];

/END OF OUTER REPEAT/

3.5.2 System Macro Commands

System MACRO commands are used in the body of the main test program and have the following general form.

`.macro-name $ iterations (pin-list);`

Where:	<code>.(period)</code>	— identifies the command as a MACRO;
	<code>macro-name</code>	— represents one of the pattern-identifying macro commands: SEQUENCE, SHIFT0, SHIFT1, ROTATE, GRAY, RANDOM, or IRANDOM.
	<code>\$</code>	— presence specifies SET LGC commands, absence specifies TEST LGC commands.
	<code>iterations</code>	— represents an optional decimal integer that specifies the quantity of SET LGC or TEST LGC command statements required in the sequence (if omitted, 1 is assumed).
	<code>(pin-list)</code>	— represents an optional list of input pins (240 max.) whose logic states will be affected by the invoked sequence of command statements. Pin specifications are separated by commas and the entire list is enclosed between parentheses. If omitted, the current pin-list associated with the particular macro-name is assumed. The current (previously specified) pin-list is determined by the translation (physical) order of the statements in the source-language program.
	<code>;(semicolon)</code>	— Terminates the macro command statement.

NOTE

Sharp symbols (`#`), signifying “all but”, must not be used in the pin-lists of macro command statements. However, symbolic names (previously defined by ASSIGN PNAME commands) and hyphens (meaning “thru”) are permitted.

Occasionally, it may be desirable to use a system macro command as an argument in a macro command itself. The following commands are the only ones that can be successfully passed as arguments to another macro:

```
.GREY(Pin-List);  
.IRANDOM(Pin-List);  
.RANDOM(Pin-List);  
.SEQUENCE(Pin-List);  
.SHIFT0 (Pin-List);  
.SHIFT1(Pin-List);
```

Use of any other macro commands as arguments to a macro command could cause unpredictable system operation.

Some examples of *valid* macro calls are:

```
.'XXX' ".ROTATE(1,2,3);";  
.'YYY' ".GREY(1,2);";
```

Some examples of *invalid* macro calls are:

```
.'XXX' ".REPEAT 4;";  
.'YYY' ".ZZZ;";  
.'AAA' ".CASE UNTIL(3=HIGH);";
```

The SET LGC or TEST LGC command statements in the sequence created by a system macro command do not include the logic states expected at the active output pins. These output expectations can be added by the CAPS simulator on CAPS/APG-equipped systems or by performing EDITing operations on the source program produced by the macro processor option of the TRANSLate mode (exists as file-name .ET). An IGNORE LGC command concerning the outputs can also be employed.

The SEQUENCE Macro

The functional process associated with the SEQUENCE macro is the binary addition of 1. By relating the low-level logic state L to binary 0 and the high-level logic state H to binary 1, the binary value of the input logic states specified by each TEST LGC command statement resulting from the use of the SEQUENCE macro is 1 higher than the preceding value. This process of binary addition, the positional relationship of pins in the pin list, and other techniques concerning the use of the SEQUENCE macro are illustrated in the following annotated examples.

```
10:      $ IL(#);
20:      .SEQUENCE (1,2,3,4);
```

In the above program segment, statement 10 sets all logic inputs low. Statement 20 is a SEQUENCE macro that calls for one (iteration default) TEST LGC command with pins 1,2,3 and 4 as the active input pins. Moreover, the binary value of these active pins will be 1 higher than previously. Since statement 10 set all input pins low, the 1 added by statement 20 produces a binary value of 0001 for pins 4,3,2, and 1 respectively.

NOTE

The first (left-most) pin specified in the pin list corresponds to the least-significant digit of the binary value while the last (right-most) pin specified corresponds to the most-significant digit.

The positional order of the pins in the pin list of statement 20 correspond to the digits of the binary value such that pins 4, 3, and 2 will remain low and pin 1 will be set high by the resultant TEST LGC command statement.

```
30:      .SEQUENCE 3;
```

Statement 30 (above) specifies an iteration of 3 and does not include a pin list. Adding this statement to the program segment produces 3 more TEST LGC commands with the binary value of the established input pin list (pins 1 through 4) incremented by a binary 1 for each test. Upon completion of statement 30, the cumulative binary value of input pins 4 through 1 is 0100 respectively.

```
40:      .SEQUENCE 16(9-6);
```

Continuing the program segment with statement 40 (above) produces 16 more TEST LGC commands but this time with pins 9, 8, 7, and 6 as the active pins. The hyphen (—) is used in the pin list to signify *thru* and, although normally used in pin lists of ascending order, can be used for pin lists of descending order but only in system macro statements as in this example, where it is desired that pin 9 be the least-significant digit of the related binary value. Also, because the logic states of these pins have not been altered by this program segment since initially set low by statement 10, the first TEST LGC command produced by statement 40 will establish a binary value of 0001 for these pins (pins 6, 7, and 8 low, pin 9 high). The binary values of the input pins for each of the 16 TEST LGC commands produced by statement 40 are as follows.

TEST	INPUT PINS				TEST	INPUT PINS			
	6	7	8	9		6	7	8	9
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	1
8	1	0	0	0	16	0	0	0	0

```
50:      BRANCH 30;
```

As an illustration of the predominance of translation (or physical) order, consider the addition of statement 50 (above) to the program segment. This is a conventional BRANCH command branching to statement 30, a SEQUENCE macro without a pin list. During the course of this program segment, two pin lists were introduced; pins 1 through 4, and pins 9 through 6. However, when first encountered, pins 1 through 4 were associated with statement 30 and this is the pin list that remains associated with the statement. When BRANCHED to, each of the TEST LGC commands produced by statement 30 increments the current cumulative binary value of input pins 1 through 4. For example, in this sample program segment, the first branching operation produces binary values of 0101, 0110, and 0111.

The SHIFTO and SHIFT1 Macros

The functional process associated with the SHIFTO and SHIFT1 macros is a shifting of the current logic states of the pins specified in a pin list one pin position to the right with the insertion of a 0 (L) or a 1 (H) respectively in the left-most pin position specified in the list. The logic-state pattern resulting from this functional process is reflected in the input specifications of the TEST LGC command statements produced by these macros with the aid of the macro processor. With the exception of the functional processes involved, the SHIFTO and SHIFT1 macros are identical to the SEQUENCE macro regarding techniques of usage.

Consider the following written program segment.

```
110:    $ IL(#);
120:    .SHIFT1 3(1-8);
130:    .SHIFTO 2(1-8);
140:    .SHIFT1 3;
150:    .SHIFTO;
```

Statement 110, a standard SET LGC command, sets all input pins low. Statement 120 is a SHIFT1 macro that produces 3 TEST LGC command statements which operate on input pins 1 through 8. Because the SHIFT1 macro inserts a 1 (H) at the left end of the pin list while shifting the current states to the right, the first TEST LGC command reflects this by setting pin 1 high. Accordingly, the second TEST LGC command sets pin 2 high and the third command produced sets pin 3 high. The test program segment resulting from statement 120 would appear as follows (remember that pin states, once set, remain until changed).

```
IH(1);
IH(2);
IH(3);
```

Statement 130 is a SHIFTO macro that produces 2 TEST LGC command statements which operate on pins 1 through 8. Because the SHIFTO macro inserts a 0 (L) at the left end of the pin list while shifting the current states on place to the right, the following continuation of the test program results.

```
IH(4)    IL(1);
IH(5)    IL(2);
```

Statement 140 is a SHIFT1 macro that produces 3 TEST LGC command statements which operate on the input pins previously assigned to SHIFT1 (pins 1 through 8). Accordingly, the following continuation of the test program is produced.

```
IH(1,6)  IL(3);
IH(2,7)  IL(4);
IH(3,8)  IL(5);
```

Statement 150 is a SHIFTO macro that, by assumption, produces 1 TEST LGC command statement which operates on the input pins previously assigned to SHIFTO (pins 1 through 8). This statement, shown below, completes the test program segment produced by the initial written program segment.

```
IH(4)    IL(1,6);
```

The insert-and-shift processes of these macros becomes obvious in a truth table that shows the logic states of the input pins for each test. Such a truth table, related to the statement labels of the written program segment, follows.

Statement Label	INPUT PINS								Test LGC Sequence
	1	2	3	4	5	6	7	8	
110	0	0	0	0	0	0	0	0	
120	1	0	0	0	0	0	0	0	1
	1	1	0	0	0	0	0	0	2
	1	1	1	0	0	0	0	0	3
130	0	1	1	1	0	0	0	0	4
	0	0	1	1	1	0	0	0	5
140	1	0	0	1	1	1	0	0	6
	1	1	0	0	1	1	1	0	7
	1	1	1	0	0	1	1	1	8
150	0	1	1	1	0	0	1	1	9

The ROTATE Macro

The functional process associated with the ROTATE macro is a shifting of the current logic state of the pins specified in a pin list one pin position to the right with the current state of the right-most pin brought around to fill the void at the left-most pin in the pin list. This process is identical to that of a ring counter and is sometimes referred to as wraparound. For example, suppose the following ROTATE macro is added to the written program segment of the preceding paragraph (para. 3.5.2.2).

160: .ROTATE 5(1-8);

This statement (label 160) is a ROTATE macro that produces 5 TEST LGC command statements which operate on input pins 1 through 8. If the number of TEST LOGIC commands required (5 in this example) were not specified, only 1 would be produced. The pin list (1-8 in this example) can be omitted if the desired pins have been previously assigned to the ROTATE macro (not the case in these examples). The continuation of the preceding truth table, as a result of statement 160, follows.

Statement Label	INPUT PINS								Test LGC Sequence
	1	2	3	4	5	6	7	8	
160	1	0	1	1	1	0	0	1	10
	1	1	0	1	1	1	0	0	11
	0	1	1	0	1	1	1	0	12
	0	0	1	1	0	1	1	1	13
	1	0	0	1	1	0	1	1	14

The GRAY Macro

The functional process associated with the GRAY macro is to produce a gray code input pattern (one input-state-change occurs with each TEST LGC or SET LGC command statement of the series). As an example, the statement:

```
.GRAY 10(1-4);
```

will produce 10 TEST LGC commands with a state pattern for input pins 1 through 4 as follows:

INPUT PINS				TEST LGC SEQUENCE
1	2	3	4	
0	0	0	0	1
1	0	0	0	2
1	1	0	0	3
0	1	0	0	4
0	1	1	0	5
1	1	1	0	6
1	0	1	0	7
0	0	1	0	8
0	0	1	1	9
1	0	1	1	10

Except for the resulting gray-code input pattern, this macro can be used in the same manner previously described for the SEQUENCE macro.

The RANDOM Macro

The functional process associated with the RANDOM macro is based on a series of pseudo-random numbers. Each TEST LGC or SET LGC command statement of the series produced by the RANDOM macro specifies a state pattern for the input pins that appears totally unrelated to other state patterns of the series. For example, the following macro command will produce SET LGC commands that might specify the logic states shown in the table for the input pins.

```
.RANDOM $ 9(1-4);
```

INPUT PINS				SET LGC SEQUENCE
1	2	3	4	
1	1	0	1	1
0	1	0	0	2
0	1	0	0	3
0	0	0	0	4
1	0	1	0	5
0	1	0	1	6
0	1	1	1	7
1	0	0	1	8
0	1	1	1	9

Except for the resulting random input pattern, this macro can be used in the same manner previously described for the SEQUENCE macro.

The IRANDOM Macro

The IRANDOM macro is a RANDOM macro that always begins at the same number in the series of pseudo-random numbers. With IRANDOM macros, an identical random-input-pattern sequence can be repeated. The following is a typical IRANDOM macro statement.

```
.IRANDOM 10(1-4);
```

NOTE

IRANDOM and RANDOM macros without pin-lists share the same pin-list established by a previous RANDOM or IRANDOM macro.

3.5.3 User-Styled Macros

A user-language macro is a test language routine that is used more than once in one or more test programs to perform a given repetitive operation. By using a macro, similar operations can be performed with altered values through the use of variables. The process of defining and then using macros is explained in the following paragraphs.

If a macro is of the type that is likely to be of use in more than one test program, it should be defined and stored in an archive user-language macro file. Refer to the Archive Facility description contained in both the Operating Instructions Manual, and the CAPS Operation Manual. Refer also to the Translator description in the Operating Instructions Manual for a description of how the macro processor locates macros in order to expand them in a test program.

Defining A Macro

A macro is normally defined at the beginning of a test program. It MUST be defined BEFORE it is first called.* Suppose the following simple arithmetic routine is similar to a number of routines required for a particular test program.

```
LD 1;  
ADD REG-5;  
STO REG-5;
```

This routine can be coded once and defined in the prologue to the main test program and then be called (referenced) in the main test program wherever it is required. If for example, this routine is only similar to the routines required by the test program and for example the general register (REG-5) varies, then the general register is made a variable to be passed as an argument to the macro. The macro is coded and defined once in the prologue and is called as needed within the program.

```
.DEFINE 'TRACK'=1;  
LD 1;  
ADD REG-?1;  
STO REG-?1;  
.ENDM;
```

A macro definition begins with a DEFINE statement and ends with an ENDM statement. As shown above, a period (.) must prefix the DEFINE and ENDM statements. The name of the macro is enclosed in tic (') marks as shown above for 'TRACK'. The =1 in the DEFINE statement indicates that one variable exists within the routine. This variable which is the first and only variable occurring in the routine, is represented by ?1. To call (insert) this routine into the body of the main test program and to specify REG-5 as the applicable general register, the following macro call statement is all that is required.

```
.'TRACK' "5";
```

In any macro call such as with the above statement the period (.) is used to identify the statement as a macro processor option of the TRANSLate Mode. The 5 enclosed in quotes (") in the above statement replaces the variable ?1 in the macro.

*If a macro is called before it is defined in a program the macro processor assumes the macro has been defined and stored in one of the macro libraries and searches for it. Also, if when a macro is defined, a macro of the same name appears in a macro library an error message "ATTEMP TO REDEFINE MACRO" is generated.

1. The values that variables assume relate only to alphanumeric characters in ASCII form and are not otherwise functional. However, the plus (+) and minus (-) signs in signal numbers can not be separate variables without the numeric quantity.
2. User-assigned reference names are limited to a maximum of 6 alphanumeric characters.

To again call this defined macro elsewhere in the program, all that is required is the insertion of a macro call statement at the desired point. In the example above the macro call specified REGISTER 5. Later if REGISTER 19 is desired, the macro call

```
.'TRACK' "19";
```

is used.

The value of the variable specified in the user-styled macro can be omitted when it is unchanged from the value specified in the previous macro call. For example, if the following user-styled macro is used in the course of the main test program, and it is the next usage of TRACK following the example shown above, REG-19 will appear in the inserted sequence.

```
.'TRACK';
```

In determining previous usage of macros, the translation or physical order of the statements (rather than the execution order) is the determining factor.

The foregoing discussion has dealt with a DEFINED routine containing only a single variable. This routine can be DEFINED with three separate variables as follows.

```
.DEFINE 'TRACK2'=3 REG=2;
LD ?3;
ADD REG-?1;
STO ?REG;
.ENDM;
```

where:

TRACK2	is the name given to the macro,
=3	specifies that there are 3 variables,
REG=2	specifies that a symbolic name is used to represent a variable, and that the value for this variable is the second value passed in the list when the macro is called.

For example, to insert in the test program a routine that would LD 1, ADD REG-5, and STO REG-5; the following user-styled macro would be called.

```
.'TRACK2' "5" "REG-5" "1";
```

The first value specified in the list (5) replaces variable ?1 in the routine. The second value specified (REG-5) replaces the symbolic variable ?REG, which has been assigned the second listing position by the DEFINE statement. Similarly, the third value specified in the list (1) replaces variable ?3 in the routine. The macro processor then inserts the routine containing the values passed in the macro call into the test program.

A DEFINEd routine can contain up to 4095 variables. When macros are called that contain many variables, a special convention is used that eliminates re-specifying values that are to remain the same as previously specified. For example, suppose the routine inserted by previous example is required again but this time the calculated result is to be stored in the CONTINUE register instead of REG-5. Using the special convention, the macro call would appear as follows.

```
'TRACK2' 2="CONTINUE";
```

This macro call specifies a single value in its list. The value (CONTINUE) is equated to the second variable of the 3-variable list by the 2= convention. Bypassing the first variable with this convention and omitting the third variable imply that the values for these variables remain the same as previously specified. The following program segment is inserted into the test program by the macro processor as a result.

```
LD 1;
ADD REG-5;
STO CONTINUE;
```

User-Style Macro Considerations:

The previous paragraphs had simple numerical values passed in the macro calls. In actuality, a value passed in a macro call list can assume the form of any integral part of a test language statement. A few considerations MUST be taken into account however in handling literals and branch addresses.

Handling Literals. The variable CAN NOT appear between two tic (') marks as with literal commands as WRITE, CCOMP, SMOVE, SCOMP etc. (These all have character strings associated with them with the strings enclosed by tic marks.)

The following macro call shows a literal (ABCD) to be passed in a macro call to a previously defined macro named MACRO. Note that the tic marks are enclosed by the quotes(''). The tics CAN NOT be left in the macro definition.

```
.'MACRO' ''1'' ''ABCD'' [20] ''(1-4)'';
```

The macro definition could be as follows:

```
CORRECT:      .DEFINE 'MACRO'=4;
               ;
               LOAD ?1;
               SCOMP ?2, ?4 [?3];
               .ENDM;
```

(The literal ?2 in SCOMP is not enclosed by tics)

```
INCORRECT:    .DEFINE 'MACRO'=4;
               LOAD ?1;
               SCOMP '?2', ?4 [?3]
               .ENDM;
```

(The ?2 is enclosed by tics and will not be recognized as a variable by the macro processor and, no error is generated.)

Handling Branch Addresses. Values are passed when a macro is called by enclosing them in quotes. A few exceptions however, include: CALL, BRANCH, and error branch [] addresses which MUST all be enclosed in brackets ([]) without the quotes. In the macro call 'MACRO' in the previous paragraph, the third value passed is the branch address [20]. Note that it is NOT enclosed by quotes.

When branch labels are passed in a macro call the following rules apply depending on whether the branch destination is in the macro (called a local branch) or in the main program (called a main program branch). The rules also depend on the renumbering option.

With Renumbering turned ON: Any branch address specified in a defined macro is assumed to be a LOCAL branch.

.TP	.ET
.DEFINE 'USER';	
COMP USOPT1 <HI=.5> [10];	1: NOP;
/LOCAL BRANCH/	
BRANCH 20; /LOCAL BRANCH/	
10: DELAY 1;	COMP USOPT1 <HI=.5> [2];
20: NOP;	BRANCH 3;
.ENDM;	2: DELAY 1;
5: NOP;	4: NOP;
.'USER1';	5: NOP;
10: NOP;	6: END;
15: NOP;	
20: END;	

A branch address passed as a value in a macro call can be either a local or main program branch depending on the format used:

"20" — Passes 20 as a local branch.

[20] — Passes 20 as a main program branch.

.TP	.ET
.DEFINE 'USER1'=2;	
COMP USOPT1 <HI=.5> [?1];	1: NOP;
BRANCH ?2;	
10: DELAY 1;	COMP USOPT1 <HI=.5> [?2];
20: NOP;	BRANCH 6;
.ENDM;	2: DELAY 1;
	3: NOP;
	4: NOP;
	5: NOP;
	6: END;
5: NOP;	
/LOCAL MAIN/	
.'USER1' "10" [20];	
10: NOP;	
15: NOP;	
20: END;	

NOTE

The only way to branch from a macro to the main program with renumbering turned on is to pass the branch address as a value ([xx]). Any other method will result in translator errors.

With renumbering turned OFF the macro processor finds the proper branch address (local or main) regardless of the format used.

.TP	.ET
.DEFINE 'EXBR'=2;	
BRANCH ?1;	
1: COMP USOPT1 <HI=.5> [?2];	
BRANCH 3; /LOCAL/	
/MAIN/	
2: COMP USOPT2 <HI=.5> [100];	10: NOP;

```
3: NOP;
.ENDM;
```

```
10: NOP;
    /LOCAL MAIN/
    'EXBR' "1" "20";
15: NOP;
20: NOP;
40: NOP;
100: END;
```

Further Examples:

The following test programs clarify and expand on the previous considerations. The first program consists of 6 macros; some of which incorporate other macros in their definitions. Several simple measurement functions are performed. To test any of them, such as DCV, that function (DCV, etc) is typed on the TEC terminal. The appropriate MEAS function is performed. Hardcopy output of the result is optional. The [CALL 100] in the 'MEAS' macro is an error branch to a routine that rings the TTY bell if the measurement cannot be made on that range.

/Prologue of Main Program/

```
.DEFINE 'RO'=3;
WRITE DEV=?1 CHAR' MEASURES '?2 ?3;
.ENDM;
```

```
.DEFINE 'HCPY'=3;
COMP USOPT1 <LO=.5> [?1];
'RO' "TTY" ?2 ?3;
.ENDM;
```

```
.DEFINE 'FUNC'=4;
CCOMP?1 [?2];
SMOVE ?3, ?4;
.ENDM;
```

```
.DEFINE 'MEAS'=5;
WRITE DEV=TEC '%154%177%175';
MEAS ?1 AT (1:2) MAX=?2 FORCE [CALL ?3]
DLY=150M;
'RO' "TEC" ?1 ?5;
'HCPY' [?4] ?1 ?5;
.ENDM;
```

```
.DEFINE 'EX'=1;
COMP CON <LO=.5> [?1];
.ENDM;
```

```
.DEFINE 'ZERO';
LD 0; STO CONTINUE;
WRITE DEV=TEC '%146';
.ENDM;
```

/Main Program/

```
10: FORMAT 14, 4;
    WRITE DEV=TEC
        '%146'^^PLEASE ENTER 1 OF THE FOLLOWING FUNCTIONS^^
        ' 1) DCV^^
        ' 2) DCI^^
        ' 3) RMU^^;
    READ DEV=TEC C;
```

BRANCH 2;

```
1: COMP USOPT1 <HI=.5> [40];
    BRANCH 3;
2: COMP USOPT2 <HI=.5> [100];
3: NOP;
15: NOP;
20: NOP;
40: NOP;
100: END;
```

/The 'RO' macro is used to write/
/the contents of the CHAR register/
/to the specified device./

/If USOPT1 is off go to [?1] -/
/if not, do 'RO' to the TTY to/
/give hardcopy printout./

/The 'FUNC' macro is used to/
/compare the contents of the TEC/
/with CHAR to define which/
/function is to be performed./

/'MEAS' is the measurement/
/routine. It uses 'RO' to TEC/
/and 'HCPY' as well./

/'EX' is used to check the status/
/of the CONTINUE button./

/'ZERO' initializes the program/
/and clears the screen for each/
/test/

```

/ DCV /
20: .ZERO';
    .FUNC' "'DCV'" [30] "'MEAS DCV AT (1:2)'^'" "DEV=TEC";
24: .MEAS' "DCV" "160" [100] [28] "'VOLTS'^'";
28: .EX' [24];
    BRA 10;

/ DCI /
30: .ZERO';
    .FUNC' "'DCI'" [40] "'MEAS DCI AT (1:2)'^'" "DEV=TEC";
34: .MEAS' "DCI" "160M" [100] [38] "'AMPS'^'";
38: .EX' [34];
    BRA 10;

/ RMU /
40: .ZERO';
    .FUNC' "'RMU'" [10] "'MEAS RMU AT (1:2)'^'" "DEV=TEC";
44: .MEAS' "RMU" "10K" [100] [48] "'OHMS'^'";
48: .EX' [44];
    BRA 10;

100: WRITE DEV=TTY '%007';
    DELAY 1;
    RETURN;

END;

```

EXPANDED TEXT

The following is the program output from the macro processor (MACRO.ET).

```

1:  FORMAT 14,4;
    READ DEV=TEC C;

/ DCV /
2:  LD 0; STO CON;
    CCOMP 'DCV' [4];
    SMOVE 'MEAS DCV AT (1:2)'^', DEV=TEC;
    MEAS DCV AT (1:2) MAX=10M FORCE [CALL 8];
    WRITE DEV=TEC CHAR' MEASURES' 'VOLTS';
    COMP USOPT1 <LO=.5> [3];
    WRITE DEV=TTY CHAR' MEASURES' 'VOLTS';
3:  COMP CON <LO=.5> [2];

/ DCI /
4:  LD 0; STO CON;
    CCOMP 'DCI' [6];
    SMOVE 'MEAS DCI AT (1:2)'^', DEV=TEC;
    MEAS DCI AT (1:2) MAX=10M FORCE [CALL 8];
    WRITE DEV=TEC CHAR' MEASURES' 'AMPS';
    COMP USOPT1 <LO=.5> [3];
    WRITE DEV=TTY CHAR' MEASURES' 'AMPS';
5:  COMP CON <LO=.5> [4];

/ RMU /
6:  LD 0; STO CON;
    CCOMP 'RMU' [1];
    SMOVE 'MEAS RMU AT (1:2)'^', DEV=TEC;
    MEAS RMU AT (1:2) MAX=10K FORCE [CALL 8];
    WRITE DEV=TEC CHAR' MEASURES' 'OHMS';
    COMP USOPT1 <LO=.5> [7];
    WRITE DEV=TTY CHAR' MEASURES' 'OHMS';
7:  COMP CON <LO=.5> [6];

```

```

8:  WRITE DEV=TTY '%007';
    DELAY 1;
    RETURN;
END;

```

The following example program illustrates an interesting application of user-styled macros that not only summarizes the preceding discussion but also introduces an additional technique — the nesting of macros.

The prologue of the written test program contains two universal statement sequences DEFINED as CHECK1 and CHECK2. CHECK1 contains five variables and equates the symbolic variable REGNO to the fourth position of value lists. The CHECK1 statement sequence includes standard test-language statements, variables identified with the question-mark (?) convention, a macro repeat-loop (not a user-styled macro), and the culminating ENDM statement.

CHECK2 contains zero variables and consists of macro statements referencing the CHECK1 routine. Notice that the user-styled macro referencing statements include value lists, with each value specification enclosed between double-quote marks ("). Notice also the 4="1" convention in the value list of the first CHECK1 macro call statement. Use of this convention eliminates the need for respecifying previously specified values of variables — the value of a variable once specified is retained until specifically changed. In this instance, the 4="1" expression means that the fourth position of the value list shall have a value of 1, the skipped third position shall retain its current (previously specified) value, and the final fifth position shall have a value of 2 as specified by "2" at the end of the list.

The second CHECK1 referencing statement changes the fourth position of the value of 2 (4="2") while, by omission, leaving the third position unchanged. The value of the final fifth position of the value list is also left unchanged, by omission. A culminating .ENDM; statement completes the CHECK2 routine of the prologue.

Following the prologue, the main test program is written. The main test program begins with a RESERVE; statement and culminates with the END; statement. Although not required by this program, the RESERVE; statement is included to show its position as the first statement of a test program. The test program continues with three standard test-language statements followed by 2 macro call statements that reference the CHECK1 and CHECK2 routines DEFINED in the prologue. The CHECK1 referencing statement passes five values to satisfy the variables of the referenced routine while no values are required of the CHECK2 referencing statement. Notice that the third value passed by the CHECK1 statement ([10]) is not enclosed between double-quote marks. It is an error branch address that, in this case, refers to the END statement of the test program. If it was enclosed with double-quote marks ("10") instead of brackets, it would refer to the SET LGC (\$) statement labeled 10: in the referenced routine of the prologue. This method of handling branch addresses facilitates the label renumbering feature of the macro processor — compare to the resultant test program.

```

RESERVE 647;
.DEFINE 'CHECK1' = 5 REGNO=4;
/START CHECK 1/
LD 1;  /KEEP TRACK OF CALLS FROM EACH LEVEL/
ADD REG-?REGNO;
STO REG-?REGNO;
/START REPEAT/
.REPEAT ?5;
10: $ IL(?1,?2) [?3];
    OH (10) [10];
    .LOOP;
/END REPEAT/
/END CHECK 1/
.ENDM;
.DEFINE 'CHECK2';

```

```

/START CHECK2/
.'CHECK1' "'7" "'12" 4="'1" "'2";
.'CHECK1' "'#1" "'6, 13-220" 4="'2";
.'CHECK1' "'7" "'12" 5="'4";
.ENDM;

LD O;
STO REG-1;
STO REG-2;
ASSIGN LGC IA(1-9, 11-240)  VILA=0.0  VIHA=3.0
                        OA(10)      VOLA=0.7  VOHA=1.5;

.'CHECK1' "'#1" "'240" [10] "'5" "'1";
.'CHECK2';
10:  END;

```

EXPANDED TEST PROGRAM (.ET)

```

RESERVE 647;
LD O;
STO REG-1;
STO REG-2;
ASSIGN LGC IA(1-9, 11-240)  VILA=0.0  VIHA=3.0
                        OA(10)      VOLA=0.7  VOHA=1.5;

/START CHECK1/
LD 1;  /KEEP TRACK OF CALLS FROM EACH LEVEL/
ADD REG-5;
STO REG-5;
/START REPEAT/
1:  $IL (#1-240) [10];
   OH (10) [1];
/END REPEAT/
/END CHECK1/
/START CHECK2/
/START CHECK1/
LD 1;  /KEEP TRACK OF CALLS FROM EACH LEVEL/
ADD REG-1;
STO REG-1;
/START REPEAT/
2:  $IL (7-12) [10];
   OH (10) [2];
3:  $IL (7-12) [10];
   OH (10) [3];
/END REPEAT/
/END CHECK1/
/START CHECK1/
LD 1;  /KEEP TRACK OF CALLS FROM EACH LEVEL/
ADD REG-2;
STO REG-2;
/START REPEAT/
4:  $IL (#1-6, 13-220) [10];
   OH (10) [4];
5:  $IL (#1-6, 13-220) [10];
   OH (10) [5];
/END REPEAT/
/END CHECK1/
/START CHECK1/

```

```
LD 1; /KEEP TRACK OF CALLS FROM EACH LEVEL/  
ADD REG-2;  
STO REG-2;  
/START REPEAT/  
6: $ IL (7 -12) [10];  
OH (10) [6];  
7: $ IL (7-12) [10];  
OH (10) [7];  
8: $ IL (7-12) [10];  
OH (10) [8];  
9: $ IL (7-12) [10];  
OH (10) [9];  
/END REPEAT/  
/END CHECK1/  
10: END;
```


SECTION 4

PROGRAM CONSIDERATIONS

4.1 General

4.2 Program Considerations

4.1 GENERAL

In order to prepare a test program, the programmer must not only have a good working knowledge of the test language, as described in Sections 2 and 3, but also have a working knowledge of both the device to be tested and the capabilities of the test system. A 1792 or 1795 test system, as shipped by GR, represents a basic system plus the addition of user selected options which determine the overall capability of the system. The remainder of this section lists some of the basic program considerations which should be observed when writing any test program.

4.2 PROGRAM CONSIDERATIONS

Program considerations vary somewhat, depending upon the type of program statements which are to be written. Four types of programming are considered in this section: power supply, logic test, analog test, and passive (resistance or continuity measurement) test.

Device Power (Fixed) [1792 A/B/D Test Systems]

1. Fixed (non-programmable) DUT power (SYS+5 and, on 1792D, SYS+15, SYS-15) is active whenever system power is turned on.
2. Test program statements cannot control fixed device power supply operation.

Universal Power Supply (UPS) Programming [1792 C/D Test Systems]

1. Types and number of power supply units.
2. Where the power supply units are installed determines the PS- number, as follows:

PS-1	PS-2	PS-3	
PS-4	PS-5	PS-6	(1792 C/D Option)
PS-7	PS-8	PS-9	(1792 D Option)

NOTES

1. The basic 1792C/D Test System includes one 10 V, 20 V, 40 V, or 80 V power supply unit and one UPS box. All remaining power supply units and UPS boxes are optional.
2. An 80 V power supply uses two UPS slots. Program the supply using the left-hand slot number only. All other power supply types use only one slot.
3. Delays encountered in charging DUT capacitance; charging rate is limited by programmed current or current limit.
4. Active loads. Power supply units can source or sink current, regardless of programmed voltage polarity.

Logic Test Programming:

1. Number of driver/sensor pins available.
2. Optional programmable test equipment available (TMU, PGEN).
3. Relay drivers.
4. Device adaptor wiring.
5. Required supply voltages, polarity, current limits, etc., for DUT power.

Power Supply Scanner [1792C/D Test Systems]

A few precautions must be observed when programming the power supply scanner option. The scanner relay contacts can be damaged by more than 1A of current. This condition is possible with the 10V and 20V power supplies which can deliver more than 1A. A similar condition can also exist when 40V and 80V power supplies are used with reactive loads.

Currents in excess of 1A during switching can be easily generated with surprisingly small reactive elements. For example, Figure 4-1 illustrates a device under test (DUT) and scanner connections. Power is applied to Pin 1 in test step 65 and some tests are then made. Pin 4 is then powered for additional tests. In this example, each node is a 20 mA load bypassed with 1.0 μ F capacitors (C1 and C4).

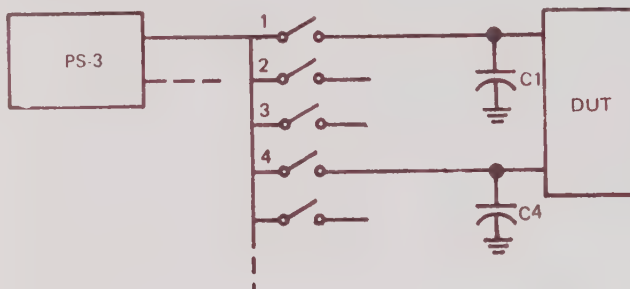


Figure 4-1. Power Supply Scanner Example

The test program for the above example would normally be structured as follows:

```
.
.
.
65:   SET PS-3 AT (1) V=15 I=.05;
.
.
.
75:   SET PS-3 AT (1,4) V=15 I=.05;
```

Note that when command 65 is executed, PS-3 charges C1; the current limit of 50 mA protects the scanner relay. However, when command 75 is executed, contact 4 closes connecting C4 (which is discharged) and another 20 mA load. Both the current of 50mA (limit) from the power supply, and current from charged capacitor C1 combine to cause a large current to flow in scanner contacts 1 and 4 until both capacitor voltages are equal. If contact failure does not immediately occur, the life of the contacts will likely be shortened.

In the above example the scanner connections can be safely made by adding a resistor (R1) from a third scanner contact to charge C4 at a limited rate as shown in Figure 4-2. The test program is then rewritten as follows:

65: SET PS-3 AT (1) V=15 I=.05;

75: SET PS-3 AT (1:3) V=15 I=.05 DLY=5M;/DLY allows C1 and C4 to charge before proceeding/

76: SET PS-3, AT (1:3,4) V=15 I=.05;

If DUT current through Pin 1 is 20mA, 30mA can charge C4 without power supply current limiting. Command 75 connects PS-3 to C4 through R1. If $R1=500\Omega$ the charging current is 30mA or less. If $R1=15\Omega$ the scanner is still protected (1A max.); however, the power supply will source current at the programmed 50mA current limit.

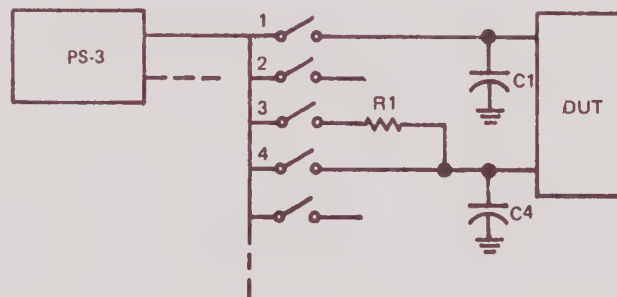


Figure 4-2. Example Limiting Charging Current

Analog Test Programming [1792D Test System]

1. Number of scanned analog DUT pins available.
2. Direct and scanned test equipment input and output signal lines.
3. Optional programmable test equipment available: ACS (function generator), DCV and DCI (DC-to-digital converter), ACV (AC-DC converter), FMU, PGEN.
4. Device adaptor wiring.
5. Required supply voltages, polarity, current limits, etc., for DUT power.

Passive Test Programming [1792D Option]

Resistance and continuity measurements require the use of the RMU option.

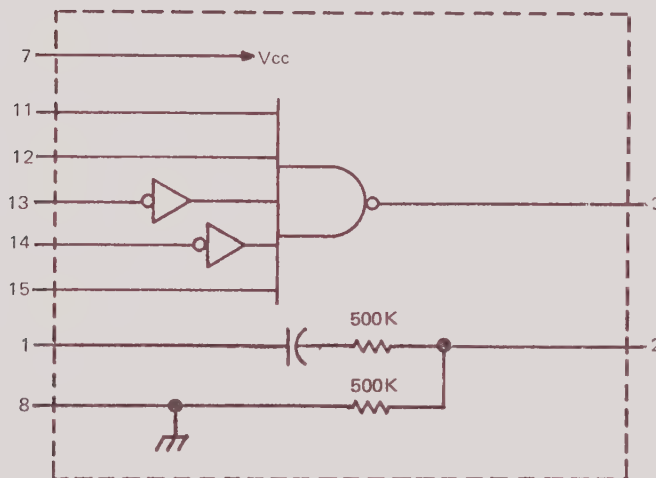
SECTION 5

SAMPLE PROGRAMS

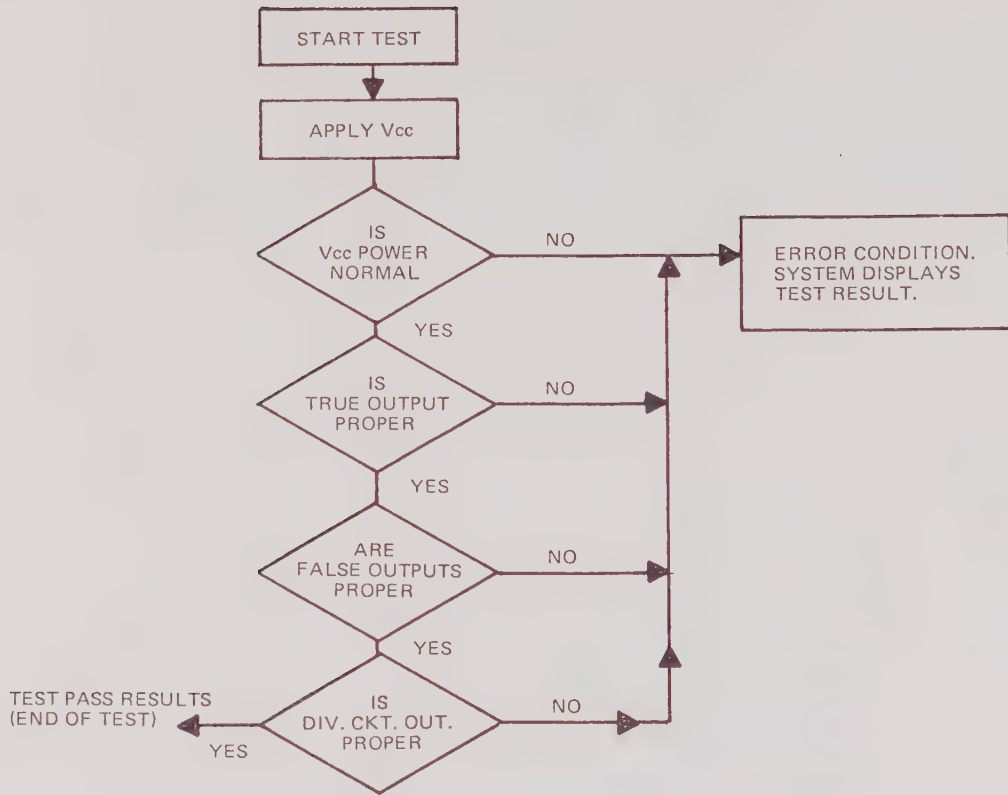
5.1 FIRST SAMPLE

A sample "device under test" (DUT) is shown below. The test objective is to test all possible logic states of the logic function shown and to determine that a voltage divider provides a required 2:1 division. Test results are displayed on the operator's control panel. If an error (failure) is detected, the data terminal displays an error message, indicating the nature of the test failure. Note that this sample program is intended to illustrate how a program can be developed using simple test language statements which control test system operation, rather than to illustrate the best method for testing a similar DUT device.

DUT Circuit



Test Steps Required



```

10: SET PS-1 AT(7:8) V=5 I=15M
20: TEST PS-1 <HI=10M LO=5M>
30: ASSIGN LGC IA (11, 12, 13, 14, 15) VILA=.5
    VIHA=4 OA(3) VOLA=.5 VOHA=3.5
40: TEST LGC IH(11, 12, 15) IL(13, 14) OL(3)
50: TEST LGC IL(11) OH(3)
60: TEST LGC IH(11) IL(12)
70: TEST LGC IH(12, 13)
80: TEST LGC IH(14) IL(13)
90: TEST LGC IL(14, 15)
100: SET ACS AT(1) F=100 V=5
110: TEST ACV AT(2) <HI=1.8 LO=1.7> SLOW
120: END;
  
```

```

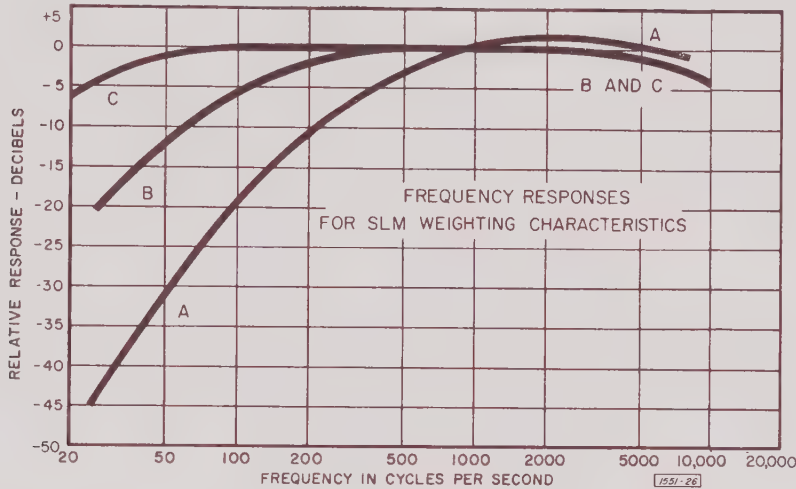
/APPLY VCC;/
/TEST FOR PROPER LOAD CURRENT;/

/DEFINE LOGIC TERMINALS AND LEVELS;/
/TEST FOR TRUE OUTPUT;/
/TEST FOR FALSE OUTPUT;/
/TEST FOR FALSE OUTPUT;/
/TEST FOR FALSE OUTPUT;/
/TEST FOR FALSE OUTPUT;/
/TEST FOR FALSE OUTPUT;/
/APPLY 3.5 VRMS TEST SIGNAL;/
/TEST FOR NOMINAL 2:1 DIVISION;/
  
```

5.2 SECOND SAMPLE

This sample illustrates the liberal use of comments for test program documentation, and demonstrates a practical application of user-defined macros for test program development.

The problem is to write a test program that checks the relative response curves of a DUT at nine selected input frequencies between 50Hz and 20kHz. The specifications of the DUT, which has three circuit outputs (A, B, and C), are shown in the following graph. A failure of any circuit to meet its specification within $\pm 3\%$ shall cause a printout identifying the circuit.



A quick analysis of the problem shows that a minimum of 64 commands are required as follows.

- The ACS shall generate the frequencies.
- The ACV shall test the circuit responses.
- Nine SET ACS commands are required, one for each frequency.
- Twenty-seven TEST ACV commands are required, one for each circuit at each frequency.
- Twenty-seven WRITE commands are required, one for each TEST ACV command in case of failure.
- One END command.

Further analysis of the problem shows that three basic commands (SET ACS, TEST ACV, and WRITE) are common requirements. By employing the macro programming aid, and user-defined macros in particular, test program development can be simplified. For example, the SET ACS command can be defined in terms of one variable, frequency, as follows:

```
SET ACS AT (INPUT) V=.707 F=?1;
```

Moreover, each frequency setting requires three TEST ACV commands — one for each circuit. Using variables for circuit output pins (?1), nominal output values (?2), failure-branching addresses (?3), and slow or fast modes because of the frequency range (?4); the TEST ACV command can be defined as follows and nested within the user-defined macro containing the SET ACS command.

```
TEST ACV ?4 AT(?1) <LIM=?2 +3% -3%> [?3];
```

Similarly, the WRITE command can be defined in one variable (failure circuit identifier) as follows and can also be nested within the user-defined macro containing the SET ACS command.

```
WRITE ^CIRCUIT' ?1 'FAILED';
```

NOTE

The up-arrow (^) is used to provide a carriage return before writing the failure message.

Consequently, only nine macro-referencing statements are required in the main driver program, each specifying the test frequency, the nominal output value expected from each circuit, and the operating mode of the ACV.

The written program followed by the resultant test program after macro processing is shown on the following pages. The main driver of the written test program starts with an ASSIGN PNAME statement that assigns the mnemonics INPUT, A, B, and C to the DUT input and output pins respectively. Next come nine macro-referencing statements which reference the user-defined macro 'FREQ' and contain all pertinent values for each frequency test. The main driver ends with the required END command.

The prologue of the written test program consists of three user-defined macros, each enclosed between DEFINE and ENDM macro commands. The first one ('AMPL') contains the basic TEST ACV command while the last one ('DIAG') contains the WRITE command. The second user-defined macro sequence ('FREQ'), which is referenced from the main driver, contains the SET ACS command and references the other two user-defined macros. Each user-defined macro is preceded by comments that describe the variables (arguments) for that macro. Notice that only two slash marks (/) are used for the multi-line comment lists with the closing slash mark located on the line following the list.

In the user-defined 'FREQ' sequence, the argument lists of the 'AMPL'-referencing macros illustrate three different methods for specifying values of variables. Values being introduced for the first time are enclosed with double-quote marks, e.g. "A", "B", "C". Values previously introduced (by the main driver in this case) and merely being relayed are specified as normal variables of a routine (without double-quote marks), e.g. ?2, ?3, ?4. Absolute branching (branching within the program segment — the user-defined 'FREQ' sequence in this case) is specified with brackets instead of double-quote marks, e.g. [1], [2], [3].

NOTE

Question marks (?) are not permitted as values for variables, i.e. "?2" is illegal and will cause an error in macro processing (translation).

The renumbering option of the macro processor was utilized to produce the resultant test program. The mixed numbering sequence of the written program is replaced by one in ascending order and the correlation of error-branching labels to target statements is maintained.

NOTE

The macro processor assigns a label each time a label is encountered in the written program. Where labeled macro-referencing statements are encountered, a label is assigned, the macro is replaced with a blank line, and the macro-invoked sequence follows. Nevertheless, a label applies until a semicolon (;), terminates a command statement.

COMMENTS PLACED OUTSIDE THE BOUNDS OF A MACRO DEFINITION
WILL NOT BE INSERTED IN THE EXPANDED TEST PROGRAM

```

/
/ ARGUMENTS FOR 'AMPL' MACRO
?1      PIN CONNECTIONS
?2      NOMINAL VALUE
?3      FAILURE ADDRESS
?4      SLOW OR FAST MODE
/

DEFINE 'AMPL' =4,
1:      TEST ACV ?4 AT(?1) <LIM=?2 +3% -3%> [?3],
        ENDM,

```

```

/ ARGUMENTS FOR 'FREQ' MACRO
?1      FREQUENCY
?2      A NOMINAL VALUE
?3      B NOMINAL VALUE
?4      C NOMINAL VALUE
?5      SLOW OR FAST MODE
/

```

```

        DEFINE 'FREQ' =5,
SET ACS AT (INPUT) V= 707 F=?1,
        'AMPL' "A" ?2 [1] ?5,
5:      'AMPL' "B" ?3 [2],
6:      'AMPL' "C" ?4 [3],
        BRANCH 4,
1:      'DIAG' " " 'A' ";
        BRANCH 5,
2:      'DIAG' " " 'B' ";
        BRANCH 6,
3:      'DIAG' " " 'C' ";
4:      NOP,
        ENDM,

```

```

/ ARGUMENTS FOR 'DIAG'
?1      CIRCUIT NAME
/

```

```

        DEFINE 'DIAG' =1,
WRITE "CIRCUIT" ?1 'FAILED',
        ENDM,

```

```

/*****
/ TEST PROGRAM STARTS /

```

```

/ ASSIGN PNAME MUST PRECEED USE OF NAMES IN STATEMENTS/
ASSIGN PNAME INPUT=11, A=3, B=5, C=6,

```

```

'FREQ' "50" " .03" " .26" " .86" "SLOW";
'FREQ' "100" " .11" " .52" " .97";
'FREQ' "200" " .29" " .79" "1.01";
'FREQ' "500" " .70" " .97" "1.02" "/FAST/";
'FREQ' "1K" "1.00" "1.00" "1.00";
'FREQ' "2K" "1.10" " .99" " .99";
'FREQ' "5K" "1.07" " .86" " .86";
'FREQ' "10K" " .75" " .60" " .60";
'FREQ' "20K" " .34" " .27" " .27";

```

```

END;

```

Written Test Program

```

/*****
/      TEST PROGRAM STARTS
/
/      ASSIGN PNAME MUST PRECEED USE OF NAMES IN STATEMENTS/
ASSIGN PNAME INPUT=11, A=3, B=5, C=6,

SET ACS AT (INPUT) V=.707 F=50,

1:      TEST ACV SLOW AT(A) <LIM= .03 +3% -3%> [4];

2:      TEST ACV SLOW AT(B) <LIM= .26 +3% -3%> [5],

3:      TEST ACV SLOW AT(C) <LIM= .86 +3% -3%> [6],
        BRANCH 7;

4:      WRITE '^CIRCUIT' 'A' 'FAILED',
        BRANCH 2;

5:      WRITE '^CIRCUIT' 'B' 'FAILED',
        BRANCH 3;

6:      WRITE '^CIRCUIT' 'C' 'FAILED',

7:      NOP,

SET ACS AT (INPUT) V= .707 F=100,

8:      TEST ACV SLOW AT(A) <LIM= .11 +3% -3%> [11];

9:      TEST ACV SLOW AT(B) <LIM= .52 +3% -3%> [12];

10:     TEST ACV SLOW AT(C) <LIM= .97 +3% -3%> [13];
        BRANCH 14;

11:     WRITE '^CIRCUIT' 'A' 'FAILED',
        BRANCH 9;

12:     WRITE '^CIRCUIT' 'B' 'FAILED',
        BRANCH 10;

13:     WRITE '^CIRCUIT' 'C' 'FAILED',

14:     NOP,

SET ACS AT (INPUT) V=.707 F=200;

15:     TEST ACV SLOW AT(A) <LIM= .29 +3% -3%> [18];

16:     TEST ACV SLOW AT(B) <LIM= .79 +3% -3%> [19];

17:     TEST ACV SLOW AT(C) <LIM=1.01 +3% -3%> [20],
        BRANCH 21;

18:     WRITE '^CIRCUIT' 'A' 'FAILED',
        BRANCH 16;

19:     WRITE '^CIRCUIT' 'B' 'FAILED',
        BRANCH 17;

20:     WRITE '^CIRCUIT' 'C' 'FAILED',

21:     NOP;

```

Resultant Test Program after MACRO Processing (Sheet 1 of 3)

```

SET ACS AT (INPUT) V=.707 F=500;

22.     TEST ACV /FAST/ AT(A) <LIM= 70 +3% -3%> [25];
23.
24.     TEST ACV /FAST/ AT(B) <LIM=.97 +3% -3%> [26];
25.
26.     TEST ACV /FAST/ AT(C) <LIM=1.02 +3% -3%> [27];
27.
28.     BRANCH 28;
29.     WRITE '^CIRCUIT' 'A' 'FAILED';
30.
31.     BRANCH 23;
32.     WRITE '^CIRCUIT' 'B' 'FAILED';
33.
34.     BRANCH 24;
35.     WRITE '^CIRCUIT' 'C' 'FAILED';
36.
37.     NOP;

SET ACS AT (INPUT) V=.707 F=1K;

29.     TEST ACV /FAST/ AT(A) <LIM=1.00 +3% -3%> [32];
30.
31.     TEST ACV /FAST/ AT(B) <LIM=1.00 +3% -3%> [33];
32.
33.     TEST ACV /FAST/ AT(C) <LIM=1.00 +3% -3%> [34];
34.
35.     BRANCH 35;
36.     WRITE '^CIRCUIT' 'A' 'FAILED';
37.
38.     BRANCH 30;
39.     WRITE '^CIRCUIT' 'B' 'FAILED';
40.
41.     BRANCH 31;
42.     WRITE '^CIRCUIT' 'C' 'FAILED';
43.
44.     NOP;

SET ACS AT (INPUT) V=.707 F=2K;

36.     TEST ACV /FAST/ AT(A) <LIM=1.10 +3% -3%> [39];
37.
38.     TEST ACV /FAST/ AT(B) <LIM=.99 +3% -3%> [40];
39.
40.     TEST ACV /FAST/ AT(C) <LIM=.99 +3% -3%> [41];
41.
42.     BRANCH 42;
43.     WRITE '^CIRCUIT' 'A' 'FAILED';
44.
45.     BRANCH 37;
46.     WRITE '^CIRCUIT' 'B' 'FAILED';
47.
48.     BRANCH 38;
49.     WRITE '^CIRCUIT' 'C' 'FAILED';
50.
51.     NOP;

```

```

SET ACS AT (INPUT) V= 707 F=5K;

43     TEST ACV /FAST/ AT(A) <LIM=1.07 +3% -3%> [46];
44
45     TEST ACV /FAST/ AT(B) <LIM= 86 +3% -3%> [47];
46
47     TEST ACV /FAST/ AT(C) <LIM= 86 +3% -3%> [48];
48
49     BRANCH 49;
50     WRITE '^CIRCUIT' 'A' 'FAILED';
51
52     BRANCH 44;
53     WRITE '^CIRCUIT' 'B' 'FAILED';
54
55     BRANCH 45;
56     WRITE '^CIRCUIT' 'C' 'FAILED';
57
58     NOP;

SET ACS AT (INPUT) V= 707 F=10K;

59:     TEST ACV /FAST/ AT(A) <LIM= 78 +3% -3%> [53];
60:
61:     TEST ACV /FAST/ AT(B) <LIM= 60 +3% -3%> [54];
62:
63:     TEST ACV /FAST/ AT(C) <LIM= 60 +3% -3%> [55];
64:
65:     BRANCH 56;
66:     WRITE '^CIRCUIT' 'A' 'FAILED';
67:
68:     BRANCH 51;
69:     WRITE '^CIRCUIT' 'B' 'FAILED';
70:
71:     BRANCH 52;
72:     WRITE '^CIRCUIT' 'C' 'FAILED';
73:
74:     NOP;

SET ACS AT (INPUT) V= 707 F=20K;

75:     TEST ACV /FAST/ AT(A) <LIM= 34 +3% -3%> [60];
76:
77:     TEST ACV /FAST/ AT(B) <LIM= 27 +3% -3%> [61];
78:
79:     TEST ACV /FAST/ AT(C) <LIM= 27 +3% -3%> [62];
80:
81:     BRANCH 63;
82:     WRITE '^CIRCUIT' 'A' 'FAILED';
83:
84:     BRANCH 58;
85:     WRITE '^CIRCUIT' 'B' 'FAILED';
86:
87:     BRANCH 59;
88:     WRITE '^CIRCUIT' 'C' 'FAILED';
89:
90:     NOP;

END;

```

Resultant Test Program after MACRO Processing (Sheet 3 of 3)

APPENDIX A INSTRUCTION SUMMARY

Command Function	Test Equipment	Terminal		Parameters		Branch
		H	L	Required	Optional	
CLEAR	ACS ACS-i ACV ALL BUS (IEC) DCI DCV FMU LGC MASCN MBSCN PASCN PBSCN PCSCN PDSCN PESCN PGEN PGEN-i PS PS-i RLY RMU SASCN SBSCN SCAN SREG (IEC) USOPT1 USOPT2 XMEAS-i XSOURCE-i					
SET	ACS ACS-i	yes	no	F = (or DC) V=	(waveshape): sine (default) TRI RECT DC (offset): OFF= (default=0 V) (mode): cont (default) GATE TRIG VCG NO	no
	BUS (IEC)				(Universal Commands): DCL LLO SPD SPE (Addressed Commands): GET GTL SDC (Addresses): MLA(n) MTA(n) UNL UNT (Secondary Commands): MSA(n)	yes

APPENDIX A INSTRUCTION SUMMARY (Cont.)

Command Function	Test Equipment	Terminal \		Parameters		Branch
		H	L	Required	Optional	
SET (Cont.)	DCI DCV	yes	yes	MAX=n RNG=n	INTG LPF NPEAK PPEAK	no
	FMU (count)	yes or (chan) A	yes (chan) B	PREC=6 TRIGA= or	TIN TINA TINB ADIR BDIR CLK= 1 thru 3 or 5 (TRIGA) or 6 (TRIGB)	no
	FMU (Fratio)	yes	yes	TRIGA= TRIGB= CLK=5 or 6	PREC=1 to 5, 7, or 8 (default=5) TIN TR ADIR BDIR	
	FMU (Freq)	yes or (chan) A	yes (chan) B	TRIGA=A or TRIGB=	PREC=1 to 5, 7, 8 (default=5) TIN TR ADIR BDIR CLK=1 to 3	
	FMU (time)	yes	yes	INT TRIGA= TRIGB=	TIN TR ADIR BDIR CLK=1 to 3 PREC=1 to 5, 7, or 8	no
	LGC (\$=SET LGC)	no	no		IH() IL() OH() OL() IST= ITT= TH() TL() TX()	no
	MASCN MASCN-i MBSCN MBSCN-i	yes	no			
	PASCN PBSCN PCSCN PDSCN PESCN	yes	no			
	PGEN PGEN-i	yes	no	TD= TS= VD= VS=	STROBE TØ (default=40N) S= (default=500 V/µs) CONT or N=(default=1)	no
	POWER	no	no	ON or OFF	LMTCHK=OFF or ON	
	PULSE -n (logic pin no)	no	no		number of ITT= pulses IST= (default=1)	
	PS-i	(optional)		V= I=	IL= (default=VL) DL Y=(default=50M)	yes

APPENDIX A **INSTRUCTION SUMMARY (Cont.)**

Command Function	Test Equipment	Terminal		Parameters		Branch
		H	L	Required	Optional	
SET (Cont)	RLY	no	no	C () and/or O ()		no
	SASCN SASCN-i SBSCN SBSCN-i	yes	no			
	SYNC —n (pulse output pin)	no	no		number of pulses (default=1)	
	TMU	yes*	yes*	VF= VS= ±		
	USOPT1 USOPT2	no	no			
	XMEAS-i XSOURCE-1	yes	yes			
TEST or MCOMP	LGC	no	no		IH() IL() OH() OL() TH() TL() TX() ITT= IST=	
ASSIGN	LGC	no	no	At least one of the following: OA() IA() OB() IB()	VOHA= (if OA is used, VOLA= these must be used) VIHA= (if IA is used, VILA= these must be used) VOHA= be used) VOLA= (if OB is used, VOHB= these must be used) VOLB= be used) VIHB= (if IB is used, VILB= these must be used) VOHB= be used) VOLB=	
REASSIGN	LGC	no	no	I() and/or O()		
IGNORE	LGC	no	no	()		no
TEST or MEAS	ACV	yes	no	AT() <> for TEST	DLY FORCE LPF LOWZ MAX=n RNG=n SLOW	yes
	COUNT	no	no	<> (for TEST)		
	DCI DCV	yes	yes	AT() <> for TEST	DLY FORCE INTG LPF MAX=n PRESET RNG=n ZERO (DCV only)	yes

*Refers to Start Pin and Stop Pin, not Scanner connections.

APPENDIX A

INSTRUCTION SUMMARY (Cont.)

Command Function	Test Equipment	Terminal		Parameters		Branch
		H	L	Required	Optional	
MEAS or TEST	DCVIF	yes	yes	I= <> for TEST	DLY FORCE INTG LPF MAX=n	yes
	DCVPRB	no	no	<> (for TEST)		yes
	FRATIO	no	no	<> (for TEST)		yes
	FREQ	no	no	<> (for TEST)		yes
	PSV-i	no	no	<> (for TEST)		yes
	RMU	yes	yes	<> for TEST	DLY FORCE INTG LPF MAX=n NEGATIVE	yes
	TIME	no	no	< > (for TEST)		yes
	TMU	no	no	< > (for TEST)		yes
BEGIN	CAPS			(file-name)		
	RECORD					
FINISH	CAPS					
	RECORD					
CHECK (IEC Bus)	BUS					yes
	SBIT OFF			Source-1 Source-2		yes
	SBIT ON			Source-1 Source-2		yes
S MOVE (IEC Bus)				Source Sink		no
S COMP (IEC Bus)				Source-1 Source-2		yes
S TEST (IEC Bus)				Source-1 Source-2		yes

Arithmetic Commands

Command	Operand	Function
LOAD or LD	Register or Value	(Register) → (AR)
STORE or STO	Register or Value	(AR) → (Register)
ADD	Register or Value	(AR) + (Register) → (AR)
SUB	Register or Value	(AR) - (Register) → (AR)
MPY or MULT	Register or Value	(AR) • (Register) → (AR)
DIV	Register or Value	(AR) ÷ (Register) → (AR)
SQRT		$\sqrt{(AR)} \rightarrow (AR)$
SQUARE		$(AR)^2 \rightarrow (AR)$
ABSOLUTE		(AR) → (AR)
ATEST	Register	Test (Register) against command limits
COMP	Register	Compare (Register) against command limits.
SIN		SIN (AR) → (AR)
COS		COS (AR) → (AR)
ATAN		TAN ⁻¹ (AR) → (AR)
LOG		Log _e (AR) → (AR)
EXP		e ^(AR) → (AR)

APPENDIX A

INSTRUCTION SUMMARY (Cont).

Utility Commands		
Command	Parameters	Function
NOP		No operation
BLOCK		Mark end of core boundary
END		Mark end of test program
BRANCH	branch label	Program branch to specified label
CALL	branch label	Program temporarily branches to specified label. Program automatically RETURNS to command following CALL.
RETURN		Automatic return from CALLED program.
ERRET		Automatic branching from CALLED program to statement specified by error branch address of CALL command.
DO	number (or value)	Repeat commands between this command and next DONE command the specified number of times.
DONE		Marks end boundary of DO loop.
ESCAPE	branch label	Terminates DO loop operation. An optional branch to the specified command label, or the following command, is executed.
DELAY	time	Inserts a no operation delay of the specified time.
PAUSE	display	Test program execution halts until operator presses CONTINUE. An optional display message may be included which will be displayed during the pause.
CLEAR ALL		Clears all programmed test system hardware functions.
CCOMP	character string and branch label	Compares character string to CHAR register contents. If characters do not compare, program branches to specified label.
READ	device and real no. or char.	Read from specified device into: AR, if R (real no.), or CHAR, if C (character).
WRITE	device and message (or message source)	Write on named device the specified characters or data, from parameter field or specified register.
OPEN WRITE	device, file-name, LG, and optional file-length	creates and defines a logging file.
CLOSE		Closes the logging file and places it in the directory.
FORMAT	No. of characters, and decimal places.	Formats write data into specified number of characters, including sign, numbers, and decimal point. A comma delimits the number. A second number specifies the number of places to the right of the decimal point.
RESERVE	Number of words	Reserves the specified number of words (up to 3777g) at the top of each field following field 1 for user-generated assembly language routines. If used, RESERVE must be the first command in the test program.
EXEC	Address	Causes the program to Jump to the core address specified (within RESERVED core) and execute user's machine-coded routines.
ASSIGN PNAME	Symbolic name and pin number.	Assigns symbolic name to designated terminal.

APPENDIX B

EXECUTABLE PAL8 TECHNIQUES

INTRODUCTION

Routines written in PAL8 assembly language can be invoked during a test program by use of test-language EXECute statements. GR-supplied software permits PAL8 routines to be invoked as either overlays or subroutines, and a defined-symbols table is included to ease the task of integrating PAL8 routines into the test program.

Overlays and Subroutines — General Differences

The basic difference between an overlay and a subroutine is the core space allocated to the particular PAL8 routine. An executable overlay is limited to 10 contiguous pages of code, is brought into core when invoked, and in the process displaces other executable overlays and/or certain system I/O routines which share the same area of core. An overlay may or may not remain in core after completing its function depending on the need for the displaced routines. The I/O routines, which may be displaced by overlays, are those that handle the processing of START and KEYBOARD CONTROL pushbuttons of the control panel, status messages, run-time errors, string handling, and READ/WRITE and OPEN/CLOSE statements.

Executable subroutines, on the other hand, are brought into core along with the test program, remain in core for the duration of the test program, and locate in prescribed areas of core that otherwise would be available for the test binary (translated test program). In a test system with the minimum core complement, the only core memory field available for test binary must be partitioned by a test-language RESERVE statement to prescribe the subroutine area. In a test system with expanded memory, the RESERVE statement is optional. When used, the RESERVE statement identically partitions all core memory fields available for test binary. Without the RESERVE statement in these test systems with expanded memory, the FIELD m specification of the PAL8 assembly-language routine prescribes the entire core memory field specified (m) as a subroutine area.

One other significant difference between overlays and subroutines is the accessibility to the zero-page links, constants, and storage areas resident in the standard GR software. These zero-page links (which includes a link to a floating-point package), constants, and storage areas are always accessible to overlays. Explicit instructions during the PAL8 assembly procedure are necessary to avail a comparable set of zero-page links to subroutines.

Programming considerations resulting from the differences between overlays and subroutines are summarized in table B-1.

Defined-Symbols Table (OVERLY.PA)

A defined-symbols table, resident on DSK:* defines additional instructions available to the PAL8 assembly-language programmer for use in writing overlays and subroutines that are to be invoked by test programs. These additional instructions enable linkage between PAL8 routines and the test program, access to the test registers of the system, acquisition of the optional branching address and data-list arguments passed by test-language EXECute statements, input/output operations, and use of a floating-point processor. A listing of these additional instructions can be obtained by entering the following command strings via the TTY.

*Due to space limitations the table is located on the TEST/HDS disk. It can be moved to either the CAPS or APG disk if desired.

```

._R PIP ↓
*LPT:<DSK:OVERLY.PA ↓

```

where:



represents the carriage-return key.

The OVERLY.PA listing also indicates what the octal number content for location SYSTYP is for a given type of system. This information is useful to make a system run-time configurable. If, for example, an EXECute routine is to be used on more than one type of system and programming differences are evident between the types of systems, a branch can be constructed that is dependent on the value of SYSTYP.

Table B-1
OVERLAYS VERSUS SUBROUTINES

Overlays

Not in core initially. Called into core when needed. May or may not remain in core. (Non-resident)

Shares same core area as certain system I/O routines. Does not interfere with core space allocated to test binary.

Full accessibility to resident zero-page links.

Has single entry point (but can use first argument to select function of multi-function overlays).

Does not require preparatory mention on monitor pages of DIAGNOSIS Mode. Can be a common routine used by different test programs. Different overlays can be invoked by a single test program.

Must be a core-image (saved) file with .SV extension stored on SYS:.

Size limited to 10 contiguous pages of code.

Must be self-initializing.

Subroutines

Brought into core with test binary and remains in core for duration of test program. (Always resident)

Uses prescribed areas of core normally allocated to test binary. Does not affect overlay area used by certain system I/O routines.

Accessibility to zero-page links through explicit assembly procedure.

Can have multiple entry points.

Requires preparatory mention of subroutine file (.UR extension) on file-usage monitor page of DIAGNOSIS Mode.

Must be a core-image (saved) file with .UR extension stored on a disk (either SYS:, or DSK: thru DSK3:, as applicable).

Size limited only by core requirements of test binary.

Need not be self-initializing if retention of developed conditions is desired for future reference (actuation of AUTOM ATIC, GUIDED PROBE, MANUAL, or ABORT pushbuttons of control panel re-initializes conditions). However, permits coding for initializing added user hardware and software at time of re-start, CLEAR ALL, and FAIL (when CONTINUE ON ERROR is off).

Accessibility of Overlays and Subroutines

Prior to using test programs containing EXECute statements, the pertinent assembly-language routines must be made available as core-image (saved) files. Using conventional practice¹, assemble, load, and save assembly-language subroutines and overlays as follows.

```
.R PAL8 )  
*xxxx,xxx < dev:uuu1,DSK:OVERLY, DEV:yyy1, dev:uuu2,DSK:OVERLY, dev:yyy2 )  
.R ABSLDR )  
*xxxx$  
.SAVE dev:xxxx.zz )
```

where:)	represents the carriage-return key;
xxxx	is a one-to-six alphanumeric character file-name that is the symbolic name of the user overlay or the file-name of the user subroutines (the second xxxx entry shown in the command string under PAL8 is optional and requests a listing file when included.)
dev:uuu1	specifies the device and the one-to-six alphanumeric character file-name of a .PA file with the definition URDEFS=1. (This file, which is not required for overlays, is optional for subroutines and, when included, causes the assembly of the zero-page links in the first two pages of the subroutine field. A second definition, URFLD=integer, is included in the file to identify the subroutine field if other than the first field available to subroutines and test binary.)
OVERLY	This is the file-name of the defined-symbols table resident on DSK:
dev:yyy1	specifies the device and the one-to-six alphanumeric character file-name of the symbolic PAL8-language source file for the overlay or subroutine.
dev:uuu2,DSK:OVERLY, dev:yyy2	This and subsequent sets of three file-names is used only when assembling a subroutine program into more than one field of core. The foregoing descriptions for uuu1 and yyy1 apply to uuu2 and yyy2 respectively.
\$	represents the ALTMODE or ESCAPE key.
dev:xxxx.zz	specifies the device, file-name and extension for the saved file. For overlays, the device must be SYS:, DSK: or one of the working devices, and must have the extension .SV. For subroutines, the extension must be .UR and the device can be any disk.

¹ Refer to Chapter 3 of DEC PDP-8 Handbook Series publications *Introduction to Programming 1972* and *OS/8 Handbook 1974*.

PAL8 CODING TECHNIQUES

All EXECutable overlays must be written so as to be self-initializing as described below. Subroutines should be self-initializing if expected to operate the same way each time they are called.

Descriptions and applications of the programming aids provided in the GR-supplied defined-symbols table are contained in subsequent paragraphs.

NOTE

In writing user overlays and subroutines, the user is encouraged to utilize the symbols supplied with the system rather than absolute values; future releases are not guaranteed to preserve the current values of symbols.

Self-initializing Code

Every EXECutable overlay must be self-initializing so as to perform in the exact same manner each time that it is invoked.

Typical of self-initializing code are the initializing instructions and constants required for looping a program. In the following example, a program segment for writing 5 line feeds to the TTY, the first three instructions initialize the routine. Without them, the program segment would generate 5 line feeds during the first run and 4096 line feeds during any subsequent rerun.

```
LF = 212
INITLF,      CLA CLL
              TAD (-5
              DCA COUNT
WRTLFL,      TAD (LF
              TSF
              JMP .-1
              TLS
              CLA CLL
              ISZ COUNT
              JMP WRTLFL
              •
              •
              •
COUNT,      -5
```

Integrating EXECutable Overlays

To integrate a PAL8 routine as an EXECutable overlay, programming aids provided by the defined-symbols table are employed as follows.

```
CURFLD=OVERFL↑10
FIELD OVERFL
*OVERLY
```

PAL8 Routine

```
CDI MAJFLD
JMP I (USRRTN
```

As indicated, overlays reside in field OVERFL and begin at OVERLY. Upon completion of the overlay program routine, return to the test program is effected by JMP I (USRRTN, which is located in field MAJFLD. CURFLD is defined at the beginning of the overlay for reference within the PAL8 routine.

An overlay may have up to 10 contiguous pages (2400₈ words) of PAL8 code. Defining anything outside this area will prevent proper execution of the overlay. Therefore, use of page-zero literals, such as TAD [10, is forbidden.

Integrating EXECutable Subroutines

To integrate a PAL8 routine as an EXECutable subroutine, programming aids provided by the defined-symbols table are employed as follows.

```

CURFLD=m↑10
FIELD m
*nnnn
ENTRY1, 0
CDF CURFLD
    PAL8
    Routine
CDI MAJFLD
JMP I (USRRTN

```

Field specification m specifies the subroutine memory field. If a test-language RESERVE statement is not included in the test program, the field specification prescribes the entire field as a dedicated subroutine memory field (permitted only on test systems equipped with optional core expansion). The memory fields in which a subroutine can reside depend upon the type of test system. Refer to the OVERLY.PA listing for details on available fields for a given system type if other than the URFLD default field is desired.

The specified starting location nnnn must be within a prescribed subroutine area. When prescribed by a test-language RESERVE statement, the subroutine area extends from word location 0 upward the number of word locations specified by not beyond word location 3777. In a dedicated subroutine memory field (no RESERVE statement), the prescribed subroutine area extends from word location 0 to word location 6377.

NOTE

Page zero locations 22, 23, and 24 in every field containing a prescribed subroutine area are designated for FAC.

Upon completion of the subroutine program routine, return to the test program is effected by JMP I (USRRTN, which is located in field MAJFLD. CURFLD is defined at the beginning of the subroutine for reference within the PAL8 routine.

Subroutines may have multiple entry points because the operand of the test-language EXECute statement for invoking subroutines explicitly specifies the field and word location of entry. However, every entry point must be coded as shown in the above example for ENTRY1 (an initial address storage location followed by a CDF CURFLD).

NOTE

The page-zero links, which allow certain overlay programming aids in subroutines, are requested during assembly and subsequently incorporated into the subroutine area, consuming slightly more than 200 word locations between 0 and 377. To obtain a loader map showing exact locations consumed (non-zero) and locations available for subroutine coding (0), create a file (URDEFS.PA) containing the single instruction URDEFS=1 and enter the following command lines via the TTY (underscored characters are printed responses). The last command line, which is CTRL C, is used to abort the printout of the loader map after the first 400 word locations have been mapped.

```

.R PAL8
*MAPIT<URDEFS,DSK:OVERLY
ERRORS DETECTED: 0
LINKS GENERATED: 9

```

```

.R BITMAP
*LPT:<MAPIT
↑C

```

Additional Overlay and Subroutine Exits

Although all overlays and subroutines must end with the exit-instruction JMP I (USRRTN, other exit-instructions can be implemented in the PAL8 routine: namely branch, error, abort, and pause.

A **branch** from the PAL8 routine can be implemented when the test-language EXEcute statement includes an optional branch specification enclosed between brackets. This branch is actually a return to the test program at the point indicated by the passed branch specification. Using the instructions shown below in the PAL8 routine implements the branch exit. If a branch specification is not contained in the test-language EXEcute statement, control is returned to the statement following the JMS I ZBRANC (or JMS I (BRANCH).

Overlays¹

```
JMS I ZBRANC
```

Subroutines¹

```
CIF MAJFLD
JMS I (BRANCH
```

An **error** exit that generates an error-message printout, clears the test system, and aborts execution of the test program can be coded in the PAL8 routine as follows.

Overlays¹

```
JMS I ZHDERO
argument /octal value`> 400
```

```
HDERRF,
```

```
HDERFA,
```

Subroutines¹

```

JMS I (HDERRF
argument /octal value > 400
•
•
•
0
MQL;CLL
TAD HDERRF
CDI MAJFLD
DCA I (IOERR
CDF CURFLD
JMP I (IOERRF

```

¹ If subroutine is assembled with URDEFS=1 in same field, coding shown for overlays may be used in subroutine.

Upon execution of the error-exit coding, an error message with the following format is generated.

ERROR NO. argument (decimal equivalent of octal value)

AT TEST STEP address-of-argument, AC=nnn (octal value of AC at error).

An *abort*, which exits the PAL8 routine and terminates execution of the entire test program without generating an error message or clearing the test system, is coded as follows.

Overlays

CDI MAJFLD

JMP I (IONPS

•
•
•

Subroutines

CDI MAJFLD

JMP I (IONPS

•
•
•

A *pause*, which lights the Pause indicator and halts execution of the test program until the CONTINUE pushbutton of the control panel is operator-actuated, is coded in the PAL8 routine as follows.

Overlays¹

JMS I ZPAUSE

Subroutines¹

CIF MAJFLD

JMS I (PAUSER

Upon actuation of the CONTINUE pushbutton, the program resumes at the instruction following the pause coding shown above.

Data-List Acquisition

EXECute statements can pass one or more arguments in a data list enclosed by parentheses to the PAL8 routine. The arguments can represent octal values, floating-point values, and ASCII characters in any combination. Each argument of the data list can be sequentially accessed only once by the overlay or subroutine.

Before the first argument that actually appears in the data list can be accessed, the octal count of the total number of arguments appearing in the list must be accessed. This count, which was automatically inserted at the beginning of the data list during test-program translation, is accessed in the same fashion that an octal-value argument is accessed. Accessing this octal count, forming its twos-complement (CIA), storing the result (DCA data-count), and testing after accessing each subsequent argument (ISZ data-count) determines the end of the data list and ensures that all arguments of the list get accessed.

The substance of the argument determines the accessing technique required.

An argument containing an *octal* value is accessed as follows; the value is returned in the AC.

¹If subroutine is assembled with URDEFS=1 in same field, coding shown for overlays may be used in subroutine.

Overlays¹

JMS I ZGETIN

Subroutines¹

CIF MAJFLD
JMS I (GETINC)

An argument containing a **floating-point** value is accessed as follows; the value is returned in the 3-word floating accumulator FAC, which is readily accessible in the calling field (and, in the case of subroutines, located in page zero of all fields containing prescribed subroutine areas). The individual memory locations of FAC (in ascending sequence of location address) are LORD (low-order mantissa), HORD (higher-order mantissa), and EXP (exponent).

Overlays¹

JMS I ZGETFP

Subroutines¹

CIF MAJFLD
JMS I (GETFPN)

An argument containing alphanumeric **text** (text strings) is accessed by coding that defines the field (text-cdf) and the starting location (text-addr) of the area where the text is stored. Text characters, which are in 6-bit ASCII form² packed 2 characters per word, are accessed in character pairs (one word at a time). The coding shown below accesses the first word (first two characters) of the text-string argument. Subsequent words are accessed by employing conventional PAL8 looping techniques. The end of the text string is indicated when the accessed word is a null (all bits are zeros).

Overlays¹

CDF MAJFLD
TAD I (BINFLD
CDF CURFLD
DCA text-cdf
JMS I ZGTTXP
DCA text-addr
●
● (other coding)
●
text-cdf, 0
TAD I text-addr /access word
CDF CURFLD
DCA text-string /store word
●
●
●
text-addr, 0

Subroutines¹

CDF MAJFLD
TAD I (BINFLD
CDF CURFLD
DCA text-cdf
CIF MAJFLD
JMS I (GETTXP
DCA text-addr
●
● (other coding)
●
text-cdf, 0
TAD I text-addr /access word
CDF CURFLD
DCA text-string/store word
●
●
●
text-addr,0

¹If subroutine is assembled with URDEFS=1 in same field, coding shown for overlays may be used in subroutine.

²See note concerning 6-bit ASCII format on page B-11.

Test-Register Correlation

User overlays and subroutines can access most of the same test registers used by the test program (see section 2 for a table of registers). Two registers not accessible to EXECutable PAL8 routines are USOPT1 and USOPT2. Except for general registers REG-1 through REG-100 (test language), which are referenced in assembly language without the hyphen (REG1 through REG100), the symbolic reference designations of the registers are unchanged.

Test registers reside in REGFLD, which is outside of the overlay and subroutine fields (CURFLD), and accomodate data in 3-word floating-point format as follows.

<i>Register Address</i>	<i>Data</i>
register-name	Low-order mantissa
register-name +1	High-order mantissa
register-name +2	Exponent

Consequently, assembly-language coding requires the crossing of field boundaries and the management of data in 3-word floating-point format. This is eased in overlays (and subroutines if URDEFS=1 in the subroutine field at assembly) by a floating-point processor that is made available through the linkages provided by the defined-symbols table. The coding technique to utilize the floating-point processor to effect a data transfer between a test register and floating accumulator FAC is as follows.

CDF REGFLD	/access test-register field
INTP	/call floating-point processor
select { FGET I (register-name	/data from register-name to FAC
or	
direction { FPUT I (register-name	/data from FAC to register-name
FEXT	/exit from floating-point processor
CDF CURFLD	/return to overlay field

In subroutines assembled without URDEFS=1, coding similar to the following can be used to retrieve data from or store data into a test register.

<i>Retrieval</i>	<i>Storage</i>
CDF REGFLD	TAD location
TAD I (register-name	CDF REGFLD
CDF CURFLD	DCA I (register-name
DCA location	CDF CURFLD
CDF REGFLD	TAD location +1
TAD I (register-name +1	CDF REGFLD
CDF CURFLD	DCA I (register-name +1
DCA location +1	CDF CURFLD
CDF REGFLD	TAD location +2
TAD I (register-name +2	CDF REGFLD
CDF CURFLD	DCA I (register-name +2
DCA location +2	CDF CURFLD
•	•
•	•
•	•
location, ZBLOCK 3	location, (3-word, floating-point value)

Input/Output Operations

Using coding techniques as exemplified in the following, alphanumeric characters can be written to, or read from; the standard I/O devices of the system. I/O devices are selected by storing the applicable symbolic name of the device in READX (for input) or in WRITX (for output). The standard I/O devices of the system are referenced as follows.

<i>Device No.</i>	<i>Symbolic Device Name</i>	<i>System Device</i>
0	PRINT	Printing device (TTY or LPT) assigned via DIAGNosis Mode options (Write only)
0	FIRST	First interrupting device (Read only).
1	TEC	Data Terminal
2	TTY	Teletype
3	PTR	High-speed paper-tape reader (Read only)
3	PTP	High-speed paper-tape punch (Write only)
4	LPT	Line printer (Write only)
5	FILE	File (Write only) See para. 3.4.13.1
6	BUS	IEC (ASCII) Bus

The following technique reads one 8-bit ASCII character into bit positions 5 through 11 of the AC. User may read subsequent characters of a text string by employing conventional PAL8 looping techniques.

Overlay Read¹

```
TAD (device-name
CDF IOFLD
DCA I (READX
CDF CURFLD
JMS I ZREAD
DCA char-addr
•
• (other coding)
•
char-addr, 0
```

Subroutine Read¹

```
TAD (device-name
CDF IOFLD
DCA I (READX
CDF CURFLD
CIF IOFLD
JMS I (READ
DCA char-addr
•
• (other coding)
•
char-addr, 0
```

The following technique writes one 8-bit ASCII character from bit positions 5 through 11 of the AC. User may write subsequent characters of a text string by employing conventional PAL8 looping techniques.

¹ If subroutine is assembled with URDEFS=1 in same field, coding shown for overlays may be used in subroutine.

Overlay Write Character¹

```
TAD (device-name
CDF IOFLD
DCA I (WRITX
CDF CURFLD
● (code to get
● character
● into AC)
JMS I ZWRITE
```

Subroutine Write Character¹

```
TAD (device-name
CDF IOFLD
DCA I (WRITX
CDF CURFLD
● (code to get
● character
● into AC)
CIF IOFLD
JMS I (WRITE
```

The following technique writes the text string beginning at location text-addr (the PAL8 pseudo-op TEXT is employed in the example). Text strings must terminate with at least 12 bits of zero.

Overlay Write Text String¹

```
TAD (device-name
CDF IOFLD
DCA I (WRITX
CDF CURFLD
JMS I ZTEX
text-addr
●
● (other coding)
●
text-addr, TEXT /text string/
0
```

Subroutine Write Text String¹

```
TAD (device-name
CDF IOFLD
DCA I (WRITX
CDF CURFLD
CIF IOFLD
JMS I (TEX
text-addr
●
● (other coding)
●
text-addr, TEXT /text string/
0
```

NOTE

With regard to text strings, and in terms of the 8-bit octal code representation of the ASCII character set (Table B-2); characters with codes 240 through 277 and 301 through 337 are each represented internally by their least-significant six bits (01 through 77), are packed two characters per memory location, and can be assembled with the PAL8 pseudo-op TEXT as shown. Characters with codes 201 through 237 and 340 through 377 can be included in text strings by prefixing the six least-significant bits with six bits of 0 (zero). To write a text string containing ASCII codes not in the above groups, such as 8-bit octal 200 for example, it is necessary to prefix the code 200 by six bits of 0 (2 octal-bits), then an octal 10. Text strings must be terminated internally with 12 bits of 0 (4 octal-bits). The test string for 200 is thus: text addr, 0010

```
0200
0000
```

Floating-Point Processor

A floating-point processing package residing in field FPFLD, is made available to overlays and subroutines by the defined-symbols table. This package contains subroutines that perform floating-point operations via three consecutive memory locations collectively designated as the floating accumulator (FAC). The individual memory locations of FAC (in ascending sequence of location address) are LORD (low-order mantissa), HORD (high-order mantissa), and EXP (exponent). The floating-point processor, which is called using the symbolic name INTP, recognizes the following mnemonic pseudo instructions. For descriptions of these instructions, refer to Digital Equipment Corp., PDP-8 Handbook Series publication "Introduction to Programming", Chapter 8, Floating-Point Packages.

¹ If subroutine is assembled with URDEFS=1 in same field, coding shown for overlays may be used in subroutine.

NOTE

The respective descriptions for FINP and FFLT are as given for FIN and FLOT in the referenced publication.

FEXT	FSQU	FSQR	FSIN	FCOS
FATN	FEXP	FLOG	FNEG	FINP
FOUT	FFIX	FFLT	FJMP	FADD
FSUB	FMPY	FDIV	FGET	FPUT
FNOR	FCDF	FSWO	FSW1	FJMS

The floating-point processor does not change the data field unless instructed to do so by an FCDF. Without an FCDF, the data field remains unchanged (as specified by the most recent CDF). For example:

$$\left. \begin{array}{l} \text{CDF REGFLD} \\ \text{INTP} \\ \text{FADD I (REG2)} \\ \text{FEXT} \\ \text{CDF CURFLD} \end{array} \right\} = \left. \begin{array}{l} \text{INTP} \\ \text{FCDF REGFLD} \\ \text{FADD I (REG2)} \\ \text{FCDF CURFLD} \\ \text{FEXT} \end{array} \right\} = \left. \begin{array}{l} \text{INTP} \\ \text{FCDF REGFLD} \\ \text{FADD I (REG2)} \\ \text{FEXT} \\ \text{CDF CURFLD} \end{array} \right\}$$

NOTES

1. A CDF is much faster than an FCDF.
2. FEXT updates all FACs page zero locations (22, 23, and 24) in all fields containing prescribed subroutine areas (see Integrating EXECutable Subroutines, page B-5).

Floating-point I/O is via the devices specified in READX and WRITX. If FLPOPT $\neq$ 0,* input is echoed on device specified by WRITX. To use FINP and FOUT for user-written I/O routines the following applies; Set FLPIPT and FLPOPT to the address of the user-written input and output routines respectively, set FPCHI1 and FPCHO1 so as to contain a CIF to a field containing the respective user-written read and write routines. When reading floating-point numbers, the last (invlaid) character is stored at FLPLCR.

NOTE

FLPIPT, FLPOPT, FPCHI1 and FPCHO1 are restored upon normal system return. However, if standard routines are to be used later by the same EXEC call, FLPIPT, FLPOPT, FPCHI1, and FPCHO1 must be restored first (implies and requires that they be saved prior to altering them).

*FLPOPT, FLPIPT, FPCHI1 and FPCHO1 are locations in field FPFLD.

Integrating Initialization Control

Standard system hardware and software are re-initialized by any of the following events: when the START button is depressed to (re)start a test program; when a CLEAR ALL statement is executed; when a END statement is encountered; or when failure is encountered in DIAGNOSIS Mode, with the Continue on Error operator control turned off. Four parameters in the defined-symbols table (CLUFLD, CLUSER, CLUSEX, IFETCH) permit subroutines to include code for also initializing user software and hardware at those times. To integrate initialization control, begin the executable subroutine with single-pass coding that sets the data field to CLUFLD and loads CLUSER and CLUSEX with the location of the initialization code. The initialization code follows JMP I (USRRTN (the normal return from a subroutine) and is terminated JMP I (IFETCH. These procedures are illustrated by the following example:

```

CURFLD=URFLD↑10
FIELD=URFLD
*400
/LOAD POINTERS INTO CLUSER AND CLUSEX
BEGIN,      0                      /STANDARD SUBROUTINE ENTRY
           CDF CURFLD              /FORMAT
BEGIN1,     TAD (CDI CURFLD
           CDF CLUFLD              /SET DATA FIELD TO POINTER FIELD
           DCA I (CLUSER
           TAD (abcd               /abcd = start location of initialization code
           DCA I (CLUSEX
           CDF CURFLD
           TAD (JMP ENTRY
           DCA BEGIN1
```

/START NORMAL SUBROUTINE
ENTRY,



```

CDI MAJFLD      /normal return
JMP I (USRRTN   /from subroutine.
```

/START INITIALIZATION CONTROL CODING
abcd,



```

CDI MAJFLD      /return from initialization
JMP I (IFETCH   /control coding.
```

NOTE

Only one executable subroutine per test program may contain initialization control coding.

Executable overlays and subroutines can be debugged using ODT. However, the CTRL/C protection of the monitor must be temporarily removed and a procedure must be followed to establish and move the ODT breakpoint into the PAL8 routine. All this is accomplished as follows.

NOTE

The following procedures require the use of absolute values for core locations and contents.

System VIII.3

```
└ GET SYS: monitor.SV↓  
└ ODT↓  
00204/1375 7410↓  
↑C  
└ SAVE SYS: monitor↓
```

where the name for the monitor depends on the software release and system type as follows:

For System VIII.3 release systems, monitor is:

MNAPG.SV (for any system with APG)
MONITR.SV (for all CAPS-equipped systems)
MNLCP.T.SV (for 1795/1799 systems)
MNITR.SV (for all TEST-equipped systems)

Whether debugging overlays or subroutines, establish a breakpoint as follows.

```
└ GET SYS:name.SV↓  
└ ODT↓  
  
07577B↓  
7576/xxxx nn↓  
200G↓
```

where: name is either CAPS,APG,TEST,1795,or HDS (for hard-disk executive) as applicable to the test system;
xxxx represents the unknown contents of core;
nn = 16 for 1792-A/B 152 for 1795/1799
 11 for 1792-C/D 10 for 1796

When the Monitor page appears on the screen of the display terminal, use normal operating procedures to update and run the DIAGNOSIS Mode. When ODT breaks at 7577 (indicating that the interpreter is in core), use the following procedure to move the breakpoint into either a subroutine or an overlay.

To move the breakpoint into a subroutine or overlay area under System VIII.3 proceed as follows:

1. Move breakpoint by typing nnnnnB and then letter C to continue (where nnnnn = the absolute value of USERBP as specified in OVERLY.PA

2. Press START pushbutton on control panel to begin test program (ODT breaks when an EXEC statement is decoded).

NOTE

As shown below, the contents of the AC indicates the type of EXECute statement encountered.

<i>AC Contents</i>	<i>EXEC Statement</i>
6203	Subroutine
6213	Overlay

3. Finally, move breakpoint to desired location in subroutine or overlay and commence debugging.

NOTE

The breakpoint may be left in a subroutine but should always be moved back to USERBP before leaving an overlay.

Table B-2
ASCII¹ CHARACTER SET

Character	8-Bit Octal	6-Bit Octal	Character	8-Bit Octal	6-Bit Octal
A	301	01	!	241	41
B	302	02	"	242	42
C	303	03	#	243	43
D	304	04	\$	244	44
E	305	05	%	245	45
F	306	06	&	246	46
G	307	07	'	247	47
H	310	10	(	250	50
I	311	11	)	251	51
J	312	12	*	252	52
K	313	13	+	253	53
L	314	14	,	254	54
M	315	15	-	255	55
N	316	16	.	256	56
O	317	17	/	257	57
P	320	20	:	272	72
Q	321	21	;	273	73
R	322	22	<	274	74
S	323	23	=	275	75
T	324	24	>	276	76
U	325	25	?	277	77
V	326	26	@	300	
W	327	27	[	333	33
X	330	30	\	334	34
Y	331	31	]	335	35
Z	332	32	↑(Λ) ²	336	36
0	260	60	←(-) ²	337	37
1	261	61	Leader/Trailer	200	
2	262	62	LINE FEED	212	
3	263	63	Carriage RETURN	215	
4	264	64	SPACE	240	40
5	265	65	RUBOUT	377	
6	266	66	Blank	000	
7	267	67	BELL	207	
8	270	70	TAB	211	
9	271	71	FORM	214	

¹An abbreviation for American Standard Code for Information Interchange.

²The character in parentheses is printed on some Teletypes.

APPENDIX C

CAPS-RELATED TECHNIQUES AND COMMANDS

SCOPE

This Appendix lists some of the methods that can be used to prepare test programs for execution under CAPS. It includes a brief description of the command set needed to define a complete CAPS section (BEGIN CAPS 'NAME' / FINISH CAPS), and the command set needed to define a CAPS Recording Section (BEGIN RECORD / FINISH RECORD) within a CAPS section. It also includes a more detailed description of those special purpose commands that are available for use within CAPS recording sections, together with some useful methods of utilizing standard test program commands both inside of CAPS recording sections and outside of these sections to accomplish specific CAPS related tasks.

CAPS SECTIONS

The portion of a test program intended for CAPS processing is referred to as a CAPS section. For complex boards, it is sometimes convenient to have separate CAPS sections for testing individual functions or partitions. A PREPARE Mode run is necessary for each CAPS section. Files prepared for a given CAPS section are prefaced with that section name.

Each CAPS section must be bounded by a BEGIN CAPS 'section-name' and FINISH CAPS command as defined in Para. 3.2.5. The BEGIN CAPS 'section-name' command also implies a BEGIN RECORD command and therefore allows the recording of failing test data in the DIAGNOSIS Mode together with the analysis of that data based on the associated CAPS data base produced during a PREPARE Mode run. The FINISH CAPS command terminates a CAPS section. Each complete CAPS section must include an ASSIGN LGC statement together with those test language commands needed to exercise the device (or portion of a device) to be simulated by CAPS. Only Input patterns are required by CAPS as Outputs are added automatically (digitally autoprogrammed) during the CAPS PREPARE Mode run. The following general guidelines further define CAPS sections:

1. The BEGIN CAPS 'section-name' **must** be the **same** as the file-name assigned to the network description (.SR file) created for the CAPS PREPARE Mode run.
2. Multiple CAPS sections may be included within the same test program, if desired. When used each BEGIN CAPS 'section-name' **must** be **different** (if two complete CAPS sections are included, then two separate .SR files must have been prepared, and each of the files must have been assigned a different file-name).
3. The BEGIN CAPS 'section-name' command implies a BEGIN RECORD command. The BEGIN CAPS command also implies a CLEAR ALL command, and thus **clears** the test system (all drivers are disconnected). An ASSIGN LGC command must therefore follow a BEGIN CAPS command so that the necessary driver/sensor pins will be connected for testing within the CAPS section. All Inputs and Outputs are assumed **low** by the simulator prior to the start of the testing sequence that follows the BEGIN CAPS, ASSIGN LGC commands. In addition, it is assumed that device power has been applied prior to the first SET or TEST LGC command in the CAPS section.
4. The FINISH CAPS command terminates the CAPS section and closes the DIAGNOSIS Mode CAPS recording file. The FINISH CAPS command also implies a FINISH RECORD command.

5. The FINISH RECORD command may be used within a CAPS section to provide a *break* in the CAPS simulation/diagnostics process so that non-CAPS related commands can be used. The command differs from BEGIN CAPS in that it does *not* clear the test system. The states of driver/sensor pins remain as they have been programmed, however, the CAPS simulator is temporarily disconnected so that non-CAPS related commands can be inserted into a CAPS section, if needed.

6. The BEGIN RECORD command re-activates the CAPS simulator/diagnostics process previously disconnected by the FINISH RECORD command.

7. During testing in the DIAGNOSIS Mode the CAPS AFL and Guided Probe Modes are entered only if failures are recorded *within* CAPS recording sections and no non-CAPS failures are encountered.

8. Only TEST LGC commands and *labelled* non-logic commands (if they are the first failing test) are recorded as failing tests within a CAPS recording section.

COMMAND RESTRICTIONS WITHIN CAPS RECORDING SECTIONS

The following commands are *not* permitted within CAPS recording sections:

BRANCH	CALL
ERRET	RETURN
COMP	CCOMP
MCOMP	CHECK
SCOMP	

Error branching within CAPS sections is *ignored* when using the CAPS AFL and Guided Probe Modes, but permitted when using the MANUAL Mode. It is therefore possible to place logic test statements (or labelled test statements) that contain error branches within a CAPS recording section for use in the MANUAL Mode *only*, as indicated by error branch [200] in the following example:

```
/ BEGIN CAPS 'EX';  
.  
.  
.  
$ IH (5);  
IL (5) OL(3,4) [200] /DUT SHOULD NOW BE INITIALIZED/;  
.  
.  
.  
FINISH CAPS;  
BRANCH 220;  
200: WRITE 'DUT DOES NOT INITIALIZE! CHECK IC3';  
PAUSE 1;  
220: END;
```

NOTE

Error branches and branching targets outside of CAPS recording sections are always functional regardless of the diagnostic operating mode.

It is also possible to include `BRANCH`, `CALL`, `ERRET`, and `RETURN` commands as part of another test statement or a labelled test statement within a CAPS section for execution in the *Manual Mode only*, as indicated by the following example:

```
TEST LGC IH(2,4)IL(6,8) [CALL 210];
```

DO loops inside of CAPS recording sections must *not* contain `TEST LGC` commands, but may contain `SET LGC` commands. To expand a small section of code many times, the `REPEAT/LOOP` macro programming, aid is recommended.

RUNNING MULTIPLE CAPS SECTIONS

Multiple CAPS sections can be handled in a few ways. Frequently, in the segmentation of a large board a few CAPS sections and possibly some non-CAPS sections are written. These sections can be handled either by linking them with branching, merging the .DA files of all digital sections into a new single .DA file, or working with a single test program with multiple CAPS sections. These alternatives are described below.

Branching

When separate test programs are created for each CAPS section and run through the Prepare Mode, all of the resulting .DA files can be linked using a `BRANCH 'NAMEX';` command (para. 3.4.5). To branch from segment-to-segment an appropriate branch command is inserted in each .DA file just before the `END;` command. This eliminates the need for merging files into a single .DA file, or even more cumbersome — writing a single test program and making several Prepare Mode passes (1 for each CAPS section) with a file-renaming operation required with each pass.

Merging .DA Files

Separate test programs can be created for each CAPS section. Next, each test program can be run separately in the CAPS Prepare Mode and all of the resulting .DA files can then be merged into a single .DA file. Using this approach, it is necessary to **remove** any `END;` statements that may have been left in the middle of the new .DA file as a result of the merger. Finally, the new .DA file must be retranslated to obtain the binary files needed for test purposes. In the following example two CAPS sections, NAMEA and NAMEB are merged and the resulting file is called NAMEC.

- a. The Prepare Mode is run for the two CAPS sections:

```
NAMEA.SR      NAMEB.SR
NAMEA.TP and  NAMEB.TP
```

- b. The resulting NAMEA.DA and NAMEB.DA files are then merged using PIP:

```
_R PIP
*_NAMEC.TP < NAMEA.DA, NAMEB.DA
```

- c. The NAMEC.TP file is then edited to remove the `END;` statement from the first section (NAMEA.DA).
- d. The Translate Mode is then run. The TP= parameter is set to TP=NAMEC.TP.

Multiple CAPS Sections — Single Test Program

A program with multiple CAPS sections can be written and retained as a single test program as shown below.

TEST
PROGRAM
AB.TP

```
BEGIN CAPS 'NAMEA';  
.  
.  
.  
.  
FINISH CAPS;  
  
MEAS ACV AT(3);  
.  
.  
.  
  
BEGIN CAPS 'NAMEB';  
.  
.  
.  
.  
.  
FINISH CAPS;  
.  
.  
.  
END;
```

This section is associated with
the network description
NAMEA.SR

Test of Analog or other non-CAPS
section of board.

This section is associated with
the network description
NAMEB.SR

After completing the single test program with multiple CAPS sections (AB.TP) and preparing the two network description (.SR files) required (NAMEA.SR and NAMEB.SR), the CAPS PREPARE Mode can be entered and a first run initiated with the following files specified:

N=NAMEA.SR
T=AB.TP

During the run, *all* Macro commands in the test program will be expanded however, only the NAMEA CAPS section will be digitally autoprogrammed. The resulting digitally autoprogrammed test program (NAMEA.DA) produced by CAPS must now be re-named as the .TP file so that a second CAPS PREPARE Mode run can be initiated to digitally autoprogram the second CAPS section. The CCL RENAME command can be used for this purpose as indicated by the following command string:

.REN NAMEA.TP < NAMEA.DA

Next, a second CAPS PREPARE Mode run can be initiated with the following files specified:

N=NAMEB.SR
T=NAMEB.TP

During this run, the second CAPS section (NAMEB) will be digitally autoprogrammed, and complete binary files will be produced for test purposes. This procedure can be repeated until all CAPS sections have been digitally autoprogrammed. Note that it is not necessary to run the Macro Processor a second time as all test program macro commands will have been expanded during the first run.

Finally, the DIAGNOSIS Mode can be entered and the NAMEB.TP file specified for device testing (TP=NAMEB.TP).

The following general guidelines apply when using multiple CAPS sections in a test program:

1. When multiple CAPS sections are included in the same test program, results for the computer guided probe are generated by executing *only* the CAPS section in which the *original* failure occurred. Statements before the BEGIN CAPS 'section-name' and after the FINISH CAPS commands are not used.

2. When multiple CAPS sections are included, the section name specified by the BEGIN CAPS command lists the CAPS data base that will be used for CAPS diagnosis if the first failing test in the initial run occurs inside of that section.

REGISTER USAGE

Registers may be referenced from within a CAPS recording section to specify analog values only. It is thus possible, for example, to specify voltage thresholds such as Device power supply values in a general register through use of the LD and STO commands, and then access the register later through the following command sequence:

```
BEGIN CAPS 'NAME';  
SET PS-1 V=REG-1 I=3;
```

Where the contents of REG-1 will become the voltage specification for PS-1.

Since the logic testing sequence of a program must be fully specified during the PREPARE Mode, registers cannot be used to specify control of logic states. For example, the following statement would be *invalid* under CAPS:

```
SET PULSE-1 REG-3;
```

to be valid under CAPS, the statement *must* specify the logical value explicitly as in the following example:

```
SET PULSE-1 10;
```

At a BEGIN CAPS 'section-name' command, the current register values (REG-n, analog value registers, etc.) are recorded (saved) by the system software so that the values can be restored prior to producing the probe results, thus allowing use of the registers to control operating conditions.

The USOPT1 (or USOPT2) registers may be used for example, to execute a conditional branch to an analog section of the test program only if the control panel USER OPTION 1 (or USER OPTION 2) button is in a specific state, or to execute the test program with High (or Low) Device power levels applied, etc. In the following example, the CAPS recording section will not be completed if the USER OPTION 1 indicator is ON (High state) until after performing the analog measurements indicated.

```
BEGIN CAPS 'NAME';  
.  
.  
.  
FINISH RECORD;  
COMP USOPT1 < LO=.8> [70];  
10: BEGIN RECORD;  
.  
.  
.  
FINISH CAPS;  
BRANCH 80;  
70: MEAS ACV AT(3);  
.  
.  
BRANCH 10;  
80: END;
```

The test program may also be executed with Device power set High, Low, or to Normal levels under control of the control panel USER OPTION 1 and USER OPTION 2 buttons. The test program segment that follows will execute the program once with Device power set to Normal levels if both USER OPTION lights are OFF, and will also execute the test program with Device power set High, Low, or to Normal values, depending on the following states of both USER OPTION lights:

<i>USER OPTION 1</i>	<i>USER OPTION 2</i>	<i>RESULT</i>
OFF	OFF	STOP (after running once with Normal levels applied).
OFF	ON	RUN LOW
ON	OFF	RUN HIGH
ON	ON	RUN NORMAL

Note that this particular program will continue to run in the mode selected until a failure occurs or both USER OPTION indicators are turned OFF. In addition, the DIAGNOSIS Mode OPERATOR CONTROLS Page V= Option must have been set to PROG in order for the program to function correctly. The test program segment that follows has been prepared to run on a 1792A test system and thus all three Device power supplies (PS-1, PS-2, and PS-3) will be set to the same levels simultaneously (refer to Para. 3.2.7).

10:	LD 5; STO REG-1; BRANCH 40;	Run with Normal Device power applied if <i>both</i> USER OPTION lights are ON
20:	LD 5; ADD .5; STO REG-1; BRANCH 40;	Run with High Device power applied if USER OPTION 1 is ON and USER OPTION 2 is OFF
30:	LD 5; SUB .5; STO REG-1; BRANCH 40;	Run with Low Device power applied if USER OPTION 2 is ON and USER OPTION 1 is OFF
40:	BEGIN CAPS 'NAME'; ASSIGN LGC IA(2, 4, 6) VILA=.4 VIHA=3.0 OA(1, 3, 5) VOLA=.8 VOHA=2.4 SET PS-1 V=REG-1 I=3; FINISH CAPS; COMP USOPT1 <LO=.5> [50]; COMP USOPT2 <LO=.5> [20]; BRANCH 10;	
50:	COMP USOPT2 <HI=.5> [30];	
60:	END;	

NON-RECORDING SECTION

The logic of statements outside of CAPS recording sections is not simulated by CAPS. The drivers of the CAPS simulator, when re-activated by a BEGIN RECORD command, assume the same values previously held when the FINISH RECORD command was encountered regardless of the logic statements in between. Conventional status messages of the Manual mode result from run-time failures that occur during the non-recording sections.

Logic statements are used in non-recording sections to initialize the UUT as required for its conditions to match the particular entry conditions of the CAPS simulator. Although this may be achievable with conditional initialization, a simple homing sequence using a few SET LGC commands (deterministic initialization) is recommended whenever feasible.

SPECIAL PURPOSE CAPS COMMANDS

CAPS Verification Commands (VH and VL)

The VH and VL commands can be used within CAPS recording sections to Verify that the simulator Outputs are in a particular state (High or Low) at a *given* test step *only*. The commands differ from OH and OL TEST LGC commands (which will also verify that simulator Outputs are High or Low) in that OH/OL commands are incremental while VH/VL commands are not. OH/OL commands remain in effect until changed, while VH/VL commands apply only to the test step listed. For example, the following VH command at test step 30

```
10:  IH(1, 4, 6);
20:  IL(4);
30:  IH(4) VH(2);
```

specifies that output pin 2 should be High at this particular test step.

When the CAPS Major PREPARE Mode G Option is run, those pins listed in test program VH/VL (or OH/OL) commands are verified against the simulated output patterns of the Good Network Simulation. Differences are noted in a difference report (.DF file) that has the format:

CAPS SECTION 'NAME' DIFFERENCES

TEST NO

USER LISTED INCORRECTLY

30

VH(2)

CAPS Transitional Commands (TH, TL, and TX)

The CAPS Transitional Commands (TH, TL, and TX) can be used within CAPS recording sections of the test program to define BLACKBOX Output states to the CAPS simulator. The BLACKBOX concept allows circuits that include analog or other unmodelable components to be *probed* successfully by grouping the elements together as a BLACKBOX then specifying the digital output states that are desired on the BLACKBOX output lines (High, Low, or X — unknown) through use of the TH, TL, and TX commands. Methods of using the Transitional commands, and methods of coding the BLACKBOX into the CAPS user library and network description are described further in Para. 2.5 of the CAPS Manual.

TEST GENERATION LANGUAGE (TGL)

TGL commands provide interaction between the APG simulator and the test program. The commands may be used to assemble test programs, "on-line" using the I sub-mode of the optional APG PREPARE mode. Programs may be assembled incrementally with segments either accepted or rejected, one section at a time. As the program is run, the programmer can easily determine how each test program segment affects the simulated model through use of the CAPS PRETTY File output. Test program segments may then be accepted or rejected, depending on an analysis of the results. The following commands make-up the Test Generation Language.

```
.REPEAT n UNTIL ( $\delta$ );  
 $\beta$   
.LOOP; or .STOPLOOP;  
  
.CASE UNTIL ( $\delta$ );  
  
 $\beta_1$   
.MORECASE;  
 $\beta_2$   
.MORECASE;  
 $\beta_3$   
.ENDCASE; or .STOPCASE;
```

Methods of preparing test programs with the TGL commands, and assembling the programs "on-line" using the I sub-mode of PREPARE are described in Appendix B to the CAPS/APG Operation Manual. Appendix B also describes some of the special "debugging" tools (such as Equation Generation, Connectivity Analysis, and Activity Tracking that are available through use of the optional E/T/C Mode supplied with APG-equipped test systems.

APPENDIX D

GR 1795/1799 DIGITAL AND POWER SUPPLY PROGRAMMING

D1.1 INTRODUCTION

Programming of the 1795 and 1799 Test Systems is achieved with a subset of the general test language that is similar to that used for programming 1792A through 1792D and 1796 Test Systems. Pertinent variations to this subset for the 1792 and 1799 Test Systems are covered in this appendix. Programming for the IEC (ASCII) Bus is covered in Appendix E.

D1.2 LOGIC DEVICE TESTING (1795-1799)

Either standard driver/sensors with hardware preset driver/sensor voltage thresholds and a single logic-level set, or programmable driver/sensors with voltage thresholds that can be set by statements in the test program and dual logic-level sets may be supplied. Both 1795B and 1799 test systems can be equipped with either type of driver/sensor, however, 1795A test systems are supplied with the standard hardware preset driver/sensor pins only.

D1.2.1 Standard Driver/Sensors (single logic-level set)

Both 1795 and 1799 Test Systems with this configuration test only one logic-level set using hardware-preset drive high level and sensor threshold-voltage. This means that all UUT inputs and outputs listed in an ASSIGN LGC statement will be exercised at the hardware-preset voltage levels, regardless of logic-level set assignments. For example, if a 1795/1799 encountered statement (A) below, its interpretation would be that shown by statement (B). (B).

Statement (A) — as programmed

ASSIGN LGC IA (2, 4, 6)	VILA=.5	VIHA=4.5
IB (8, 10, 12)	VILB=-4	VIHB=2.9
OA (1, 3, 5)	VOLA=1.5	VOHA=3.9
OB (7, 9, 11)	VOLB=-3	VOHB=1

Statement (B) — 1795/1799

ASSIGN LGC IA (2, 4, 6, 8, 10, 12)	VILA=.1	VIHA=3.0
OA (1, 3, 5, 7, 9, 11)	VOLA=1.3	VOHA=1.3

As shown by the above two statements, all input and output pins assigned by the program are interpreted as being in the same logic-level set. Also, the programmed high and low threshold-voltage specifications (which must be included for completeness of the command statement) do not affect the fixed voltage thresholds of the 1795/1799 test systems. Other values between +1 volt and +12 volts are available for VIHA. Similarly, other values between zero volts and +12 volts are available for VOHA (and VOLA which tracks).

The values are set through resistor selection on the logic board.

D1.2.2 Programmable Driver/Sensors (dual logic-level set)

Programming considerations for the 1795/1799 programmable driver/sensors are identical to those covered in para 3.2.6 of this manual, with the following exceptions:

Low and high logic-levels are individually programmable between -15 V and +15 V for each logic-level set.

Logic-level sets **must** be assigned to driver/sensor pins in groups of 12 pins per set. Designating both logic-level sets A, and B within a 12 pin group is **not** permitted and will result in an error message, when the DIAGNOSE mode is entered for UUT testing.

D1.3 HYBRID DEVICE TESTING (1799)

A 1799 test system may be supplied with up to 192 hybrid shelf pins for UUT testing. Pins are supplied in groups of 48 so that a test system may be equipped with 48, 96, 144, or 192 hybrid pins. Hybrid pins are unique in that they allow both driver/sensors and/or analog source and measurement devices to be scanned to shelf pins under control of the test program. It is therefore possible to utilize the same group of shelf pins for both digital and analog device testing. Methods of programming hybrid pins are similar to the methods used for 1796 test systems and are described in Para. F2.2.2 of Appendix F.

D1.4 DEVICE POWER SUPPLIES

A 1795 Test System may be equipped with two types of device power supplies; a fixed user power supply and/or triple power supplies. A 1799 Test System may be equipped with any combination of fixed, triple, or universal power supplies.

D1.4.1 Fixed User Power Supply

In the 1795 and 1799 Test Systems the Fixed power supply must always be referenced as PS-1. The programming statements for the Fixed power supply are similar to those described in para 3.2.7 for the 1792A Device Power Supplies, with the following overriding considerations.

The fixed user power supply provides three voltage outputs rated at 10 amperes maximum and, when installed, **must** be referenced as PS-1. The three nominal voltage* outputs are 4.75 V (low), 5.00 V (normal), and 5.25 V (high). The low, normal, and high outputs respectively correspond to the PS-1A=, PS-1B=, and PS-1C= ranges of the SETUP Mode Power Supply Configuration page. Normally for the fixed user power supply, this page would be setup as follows: (note, that the hundredths digit must be 5 or 0):

PS-1: 1A=4.65, 4.85
 1B=4.90, 5.10
 1C=5.15, 5.35

Then, the following SET PS-1 statement would select the low range output of the fixed user power supply because the voltage specified (V=4.7) is within the range setup for 1A above.

100: SET PS-1 V=4.7 I=10 DLY=50 [80];

As shown in the above statement, I=10 must always be specified when programming the fixed user power supply. The statement label (100:), the DLY=50 parameter, and the error branch [80], are used as described in the manual.

NOTE

The V= and V0= options of DIAGNOSE Mode also affect the selection of the power supply output. That is; when V=HIGH or V=LOW is specified on the Operator Controls page of DIAGNOSE Mode and the V0= option of the Select Miscellaneous Options page is specified as a numeric percentage, the computer accordingly calculates an adjusted voltage for the V= specification in the SET PS-1 statement. This calculated adjusted voltage is then compared to the ranges specified on the Configuration page of the SETUP Mode to select the power supply output.

D1.4.2 Triple Power Supplies (1795/1799)

In 1795 and 1799 Test Systems, the Triple Power Supply (TPS) consists of three independent sections: PS-1 (or PS-4), PS-2 (or PS-5), and PS-3 (or PS-6). Each section provides three voltage outputs. The actual voltages at the outputs are preset by hardware (resistor plugs). Each of the three voltage outputs of the first section (PS-1 or PS-4) can be preset to any voltage between 0.5 volts and 7.5 volts at 5 amperes maximum (nominal 5-volt plug supplied). Each of the three voltage outputs from the remaining two sections (PS-2 and PS-3 or PS-5 and PS-6) can be preset to any voltage between 0.5 volts and 20 volts at 1.5 amperes maximum (nominal 15-volt plugs supplied). The three voltage outputs of each section are viewed as output A, output B, and output C. Voltage ranges corresponding to the three outputs from each section are specified on the Power Supply Configuration page of SETUP Mode. Assume this page is setup as follows:

PS-4: 4A=4.65, 4.85 4B=4.90, 5.10 4C=5.15, 5.35
PS-5: 5A=13.40, 13.70 5B=14.90, 15.10 5C=16.40, 16.60
PS-6: 6A=13.40, 13.70 6B=14.90, 15.10 6C=16.40, 16.60

Then, the following command statement selects output A of the first section (PS-4) because the value of the V= specification falls within the range specified for the 4A= parameter:

110: SET PS-4 V=4.7 I=5 DLY=50 [80];

Similarly, the following command statements select outputs B and C of the second (PS-5) and third (PS-6) sections respectively:

120: SET PS-5 V=15 I=1.5 DLY=50 [80];

*Factory preset.

125: SET PS-6 V=16.40 I=1.5 DLY=50 [80];

Note that the V= specification of the command statement only selects the output of the power supply; the actual voltage at the output is preset by the resistor plug and does not necessarily reflect the V= value. The actual (nominal) voltages at outputs A, B, and C with the factory-supplied 5-V plug are 4.75, 5.0, and 5.22 ($\pm 5\%$) respectively; and with the 15-V plug are 13.66, 15.04, and 16.55 ($\pm 10\%$).

The I= specification in the command statement must be I=5 when setting PS-1 or PS-4 and must be I=1.5 when setting the other two sections of the triple power supply. The other specifications (statement label, error branch, and DLY= specification) and programmed as described in the manual.

NOTE

See preceding note (under Fixed User Power Supply).

D1.4.3 Universal Power Supplies (1799)

Programming considerations for the Universal Power Supplies (ups) are identical to those for the 1792 C/D Test System, outlined in para 3.2.7.3 of this manual, with the following exceptions:

In 1795A and 1795B test systems (which do not use UPS's) any of the fixed and TPS device power supply configurations shown in Figure D-1 may be utilized.

The 1799 (which can use UPS's) may have any of the device power supply configurations shown in Figure D-2.

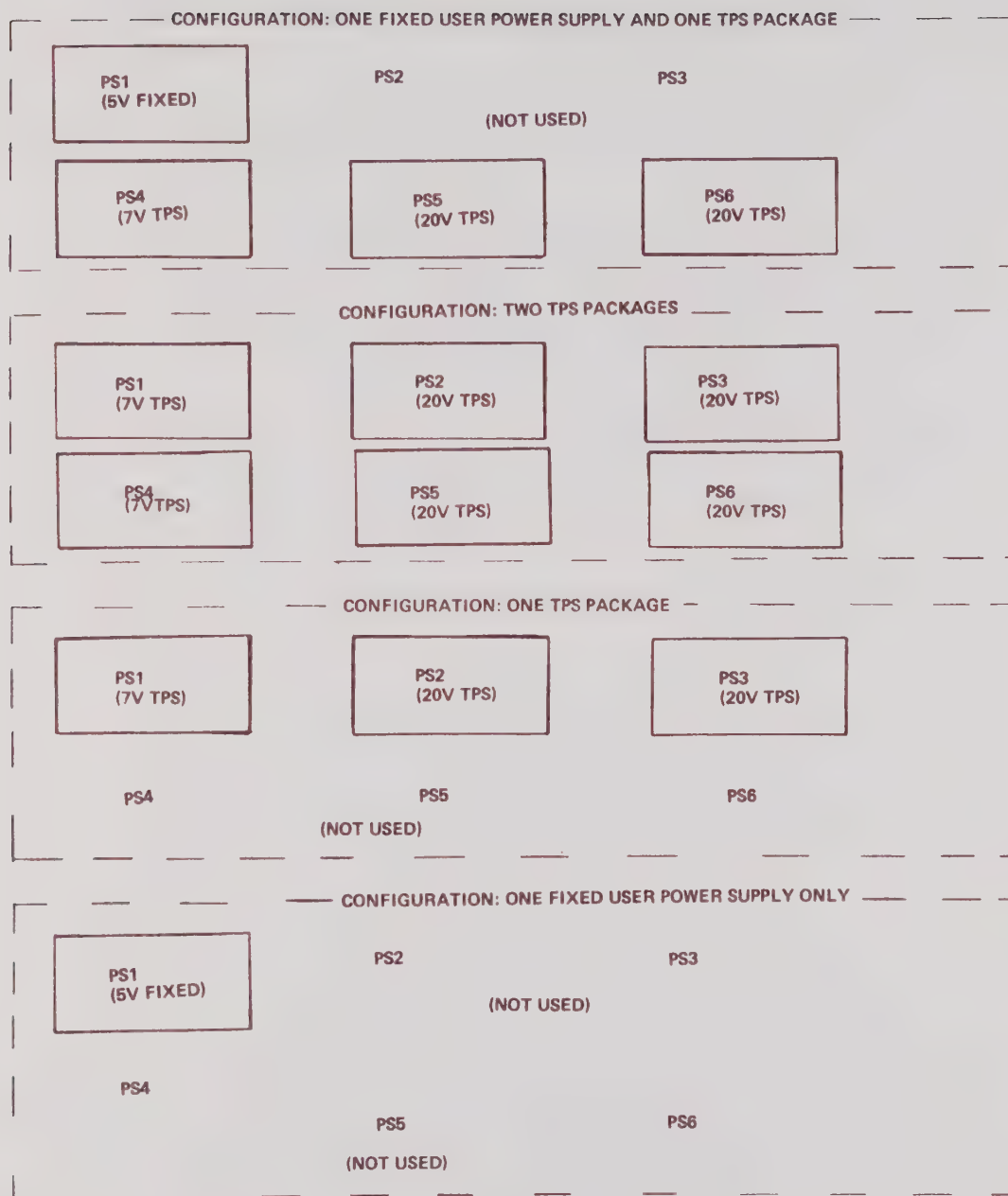
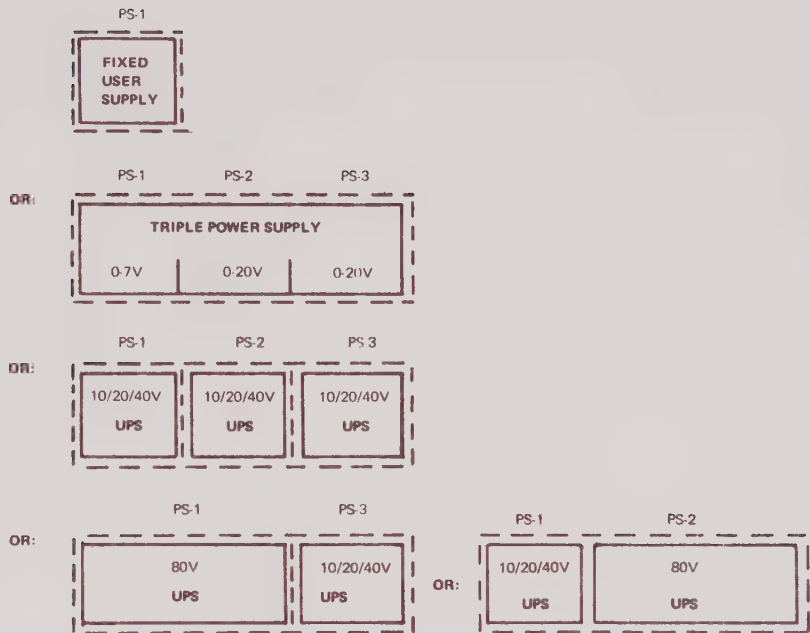


Figure D-1. 1795 Power Supply Configuration.

UPS/TPS Box # 1:



UPS/TPS Box # 2



Figure D-2. 1799 Power Supply Configuration.

APPENDIX E

IEC (ASCII) BUS PROGRAMMING

IEC BUS-PROGRAMMING

The IEC Bus option is GR's implementation of IEEE Standard 488-1975*. With the IEC Bus option, up to fifteen external devices conforming to the standard can be attached to the test system and remotely operated using high-level test-language programming. In operation, the test system acts as sole controller and can implicitly perform the functions of talker and listener. The programming required to operate a particular device attached to the bus depends on the operating characteristics of that device, and it is therefore necessary to consult the remote-programming instructions supplied with the particular device.

Position List

Basic to the operation of the IEC Bus, is the ability to manipulate, transfer, and check individual ASCII characters and symbols. A character-string register (SREG) with a capacity for 128 ASCII characters has therefore been implemented. Except for a CLEAR SREG command, which replaces all data in SREG with ASCII NUL characters, the string register is referenced by a position list in command statements. This position list, which is bounded by parentheses, explicitly identifies the ASCII-character locations (0 through 127) of SREG that are to be sequentially and individually accessed during the command. The following is an example of a position list:

(0-2, 4, 7-5, 9+)

Conventions used in a position list are the hyphen (-) which means *thru*, and the plus sign (+), which means *additional locations* (in ascending sequence) as required. In the above example, at least eight locations of SREG are accessed in the following sequence:

0, 1, 2, 4, 7, 6, 5, 9, 10, 11, . . .

Data Transfer With The IEC Bus.

To remotely program and retrieve data from external devices attached to the IEC Bus, ASCII data must be transferred via the bus. The bus, therefore, is considered an input/output device which can be referenced in data transfer commands as follows:

DEV=BUS

*Copies of IEEE Standard 488-1975 "Digital Interface for Programmable Instrumentation" may be purchased from The Institute of Electrical and Electronics Engineers, 345 East 47th Street, New York, NY 10017.

Command Repertoire

Full use of the IEC Bus is facilitated by the following repertoire of test-language command statements, shown in their general form:

```
optional-label: CLEAR BUS;  
optional-label: SET BUS [optional-error-branch] bus-command-string;  
optional-label: CHECK BUS [error-branch];  
optional-label: SMOVE source, sink;  
optional-label: SCOMP source-1, source-2 [error branch];  
optional-label: STEST source-1, source-2 [optional-error branch];  
optional-label: CHECK SBIT OFF source-1, source-2 [error-branch];  
optional-label: CHECK SBIT ON source-1, source-2 [error-branch];  
optional-label: CLEAR SREG;
```

The **error-branch**, used for conditional program branching, is required wherever shown in the foregoing repertoire and may take any of the five accepted forms typified by the following examples:

```
[300], [CALL 210], [RETURN], [ERRET], [ ]
```

The **optional-error-branch** wherever shown in the above repertoire may be specified, if desired, using any of the accepted forms shown above for error branch. When omitted and execution of the command statement calls for a branch, the system error handler will display an error message before the program proceeds to the next command statement.

The **bus-command-string** is a series of individual bus commands separated by commas. The various bus commands are listed in a subsequent discussion of controller function.

A **sink** accepts data during a data transfer and can be specified as a position list (referencing SREG), a specific register (REG-3, REG-12, etc.), or an output device (DEV=TTY, DEV=BUS, etc.).

A **source**, including **source-1** and **source-2**, furnishes data during a data transfer or comparison check and can be specified as a position list (referencing SREG), a literal string ('MORE TEXT'), a numeric constant (1.3E7 or 52.5, for example), a specific register (REG-9), or an input device (DEV=TTY, DEV=BUS, etc.).

Execution of command statements, which specify sinks and/or sources, terminates when one of the following conditions occurs:

1. The specified sink becomes full. If the sink is a position list, it is full when the last location identified is accessed and accepts a character.
2. The specified source becomes exhausted. If the source is a position list, it is exhausted when the last location identified is accessed and furnishes a character.
3. Bus time-out; if the sink or source is DEV=BUS, time-out occurs one-second after last character transaction.

Controller Function

GR's implementation of the (IEEE) standard presumes only one controller — the test system. This controller function is invoked by use of a CLEAR BUS, a SET BUS, or a CHECK BUS command.

The CLEAR BUS command, generally used to initialize the standard interface system, asserts the interface clear (IFC) signal of the bus.

The SET BUS command asserts the attention (ATN) and remote enable (REN) signals of the bus and transfers the multiline interface messages specified in the bus-command-string. Multiline interface messages supported by GR's implementation of the standard (including device addresses, universal commands, and addressed commands) are as follows:

Universal Commands:

- DCL — Device Clear
- LLO — Local Lock Out
- SPD — Serial Poll Disable
- SPE — Serial Poll Enable

Addressed Commands:

- GET — Group Execute Trigger
- GTL — Go To Local
- SDC — Selected Device Clear

Addresses:

- MLA(n) — My Listen Address (n = 0 to 30 decimal*)
- MTA(n) — My Talk Address (n = 0 to 30 decimal*)
- UNL — Unlisten
- UNT — Untalk

Secondary Commands:

- MSA(n) — My Secondary Address (n = 0 to 30 decimal*)

*The decimal address designation can be represented by a reference to a register, e.g. MLA(REG-3), MTA(AR), etc. The decimal address is then the truncated integral value of the contents of the register.

The following is a typical example of a SET BUS statement:

```
SET BUS UNL, UNT, MLA(22);
```

This statement clears all listeners and talkers from the bus and then explicitly establishes device 22 as a listener. Notice that three multiline interface messages, separated by commas, are specified in the bus-command-string of the statement. They are performed in the sequence in which they appear. The functions or operations resulting from the messages, however, depend on the operating characteristics of the devices attached to the bus. Since an optional-error-branch is not explicitly specified in the above command, incomplete execution causes the system error handler to generate a system error message.

The decimal addressing scheme, as used with MLA(n), MTA(n), and MSA(n), messages is essentially the decimal equivalent of the binary information presented over the five least-significant lines of the 8-line DIO data portion of the bus. A few address examples illustrating this scheme are shown below while an Address Code Conversion Chart (See Table E-1) is provided for a complete and comprehensive reference.

DIO Line	Decimal Equivalent
5 4 3 2 1	
0 0 0 0 0	0
0 0 0 0 1	1
1 0 1 1 0	22
1 1 1 0 1	29
1 1 1 1 0	30

For example, a manufacturer may state that a device is originally factory preset with a listen (MLA) address of 1 and a talk (MTA) address of Q (in ASCII Code). From Table E-1 it is found that this is a binary code of 10001 or the decimal equivalent 17. Thus the listen and talk addresses are specified MLA(17) and MTA(17), respectively, in a SET BUS command. Similarly, if a manufacturer specified the address in octal, the decimal equivalent is found the same way using Table E-1. It makes no difference what the octal address is (for either talk or listen), the decimal equivalent is found from the table. The differentiation between talk and listen is in the MTA and MLA parameters, not in the decimal equivalent. The same results can be obtained for an octal code without reference to the table by representing the octal value as a binary number, then using **only** the 5 least significant binary bits the decimal equivalent can be found by inspection. For example, the octal address 111 is represented as 001001001 in binary form. The 5 least significant bits (01001) represent the decimal value 9.

The CHECK BUS command, which requires an error-branch specification, checks the status of the service request (SRQ) line of the bus. If the SRQ line is active (asserted by some device attached to the bus), the program continues with the next statement following the CHECK BUS command. Otherwise (when SRQ is not active), the error-branch of the CHECK BUS command is taken. For example, the following command statement will keep looping on itself until some device attached to the bus asserts the SRQ signal:

```
20: CHECK BUS [20];
```

Talker/Listener Function

The Talker or Listener function of the test system is implicitly established whenever DEV=BUS is specified as the sink or source, respectively, in a command statement. DEV=BUS references the 8-line DIO data portion of the IEC Bus. For example: the following, a legitimate command statement using the DEV=BUS specification, implicitly establishes the test system as a talker.

```
WRITE DEV=BUS 'R7FOT1M3E';
```

The test system as a talker presents each character of the literal string enclosed between tic (') marks onto the DIO data portion of the bus in ASCII format. All currently established listeners attached to the bus read the characters as they are presented using the standard handshaking routine as implemented by hardware. The characters are transferred in the left-to-right sequence in which they appear in the literal string. The command terminates when all characters of the string have been transferred.

Table E-1

ADDRESS CODE CONVERSION CHART

ASCII CODE CHARACTER		BINARY CODE						OCTAL CODE		DECIMAL EQUIVALENT (n)
Listen Address	Talk Address	DIO Line → b ₇	5 b ₆	4 b ₅	3 b ₄	2 b ₃	1 b ₂	Listen	Talk	
SP	@		0	0	0	0	0	040	100	0
!	A	See	0	0	0	0	1	041	101	1
"	B	note	0	0	0	1	0	042	102	2
#	C	below	0	0	0	1	1	043	103	3
\$	D		0	0	1	0	0	044	104	4
%	E		0	0	1	0	1	045	105	5
&	F		0	0	1	1	0	046	106	6
'	G		0	0	1	1	1	047	107	7
(	H		0	1	0	0	0	050	110	8
)	I		0	1	0	0	1	051	111	9
*	J		0	1	0	1	0	052	112	10
+	K		0	1	0	1	1	053	113	11
,	L		0	1	1	0	0	054	114	12
-	M		0	1	1	0	1	055	115	13
.	N		0	1	1	1	0	056	116	14
/	O		0	1	1	1	1	057	117	15
0	P		1	0	0	0	0	060	120	16
1	Q		1	0	0	0	1	061	121	17
2	R		1	0	0	1	0	062	122	18
3	S		1	0	0	1	1	063	123	19
4	T		1	0	1	0	0	064	124	20
5	U		1	0	1	0	1	065	125	21
6	V		1	0	1	1	0	066	126	22
7	W		1	0	1	1	1	067	127	23
8	X		1	1	0	0	0	070	130	24
9	Y		1	1	0	0	1	071	131	25
:	Z		1	1	0	1	0	072	132	26
;	[		1	1	0	1	1	073	133	27
<	\		1	1	1	0	0	074	134	28
=	]		1	1	1	0	1	075	135	29
>	~		1	1	1	1	0	076	136	30

Note: Only the first five low-order bits of the binary code are given, and these bits are the same for both listen and talk addresses. Sixth and seventh bits determine whether address is listen (01) or talk (10) and are implicitly specified by MLA or MTA bus commands respectively.

Although the WRITE command previously described is legitimate, use of the SMOVE command is recommended, primarily because of its greater versatility. In terms of results, the following SMOVE command statement is identical to the previously described WRITE statement:

```
SMOVE 'R7F0T1M3E', DEV=BUS;
```

In the above SMOVE statement, the literal string is specified as the source and DEV=BUS is specified as the sink with a comma (,) separating the two. The test system is established as a talker and transfers the sequence of characters in the literal string in ASCII format to all currently established listeners attached to the bus.

NOTE

To insure that no more than one talker is active at any one time and that at least one listener is established, a command similar to the following should be issued before implicitly establishing the test system as a talker with the WRITE or SMOVE command statements as previously described:

```
SET BUS UNT, UNL, MLA(n);
```

Specifying DEV=BUS as the source in the SMOVE command statement implies that the test system is to perform the listener function. Accordingly, the following statement establishes the test system as a listener and allows the transfer of ASCII data from a currently established talker attached to the bus to a character sink via the 8-line DIO data portion of the bus:

```
SMOVE DEV=BUS, sink;
```

Of course, before implicitly establishing the test system as a listener with the above command, some other device on the bus must be explicitly designated a talker by a SET BUS MTA(n) command and the designated talker must be ready to transfer data. Also, the CHECK BUS command can be used to determine if a device capable of asserting SRQ has data ready for transfer.

In GenRad test systems Listen Address= 0 (MLA (0)) is always used to refer to the system itself. When MLA (0) is included in a SET BUS command the test system will immediately be established as a listener, instead of waiting for an SMOVE or a READ command before doing so.

Use of MLA (0) is recommended with SET BUS MTA(n) commands in cases where a device is ready to transfer data as soon as it is addressed to talk. During test program execution some time elapses between a SET BUS command, which selects talker and listener(s) and the SMOVE or READ command, which actually transfers data into the system. If no listener is selected during this interval, and if the talker begins a handshake, the state of the interface is such that the talker will think the handshake was successfully completed. This can result in the first few characters in an input string being lost. However, including MLA (0) in the SET BUS command selects the system as a listener before the talker can begin the first handshake; thus transfer of the input string does not actually begin until the READ or SMOVE command is executed.

Talk Address = 0 is reserved for use by the test system.

Character Manipulation and Checking Techniques

To facilitate character manipulation and checking, the character string register (SREG) is available to the programmer for use as a source or sink. This string register has 128 character positions, each of which is addressable and therefore readily accessible. The concept of a position list is used in command statements to reference the string register. The following command-statement example, where a position list is used for the character sink, best describes the concept:

```
SMOVE DEV=BUS, (0-2, 4, 7-5, 9+);
```

As shown in the above command, the position list is enclosed with parentheses. This command, which establishes the test system as a listener, transfers a sequence of ASCII characters from the DIO data portion

of the bus to the specified positions of the string register. The first three characters appearing on the bus are transferred to the first three positions of the string register (0, 1, 2). The next character appearing on the bus is transferred to position 4 of the register, leaving position 3 unaltered. Three more characters are then transferred from the bus into positions 7 through 5, which, in effect, reverses their order. All remaining characters appearing on the bus are then transferred in consecutive order to the string register beginning at position 9. This command terminates one second after no more characters appear on the bus (time out), or when position 127 of the string register has been reached and filled. If, on the other hand, fewer positions were specified than characters available from the bus, the command statement would terminate when the specified positions were exhausted.

The following command-statement example illustrates the use of a variable in a position list:

```
SMOVE (0-2, REG-3, 7-5), DEV=BUS;
```

The above command statement, which establishes the test system as a talker, transfers a sequence of ASCII characters from the specified positions of the string register to the DIO data portion of the bus. One of the positions is specified by variable REG-3. The position specified by this variable is equivalent to the current, truncated value stored in REG-3. For example; if the value of REG-3 is 13.85, the characters transferred to the bus by this command will be from string-register positions 0, 1, 2, 13, 7, 6, and 5.

The following command-statement examples are annotated to summarize *character manipulation* using the SMOVE command:

```
SMOVE REG-3, DEV=BUS;           /note 1/

SMOVE DEV=TTY, (1-5, REG-3, 19+); /note 2/

SMOVE 'LITERAL STRING', (1+);   /note 3/note 6/

SMOVE REG-5, REG-8;             /note 4/

SMOVE DEV=BUS, DEV=TTY;         /note 5/
```

NOTES

1. This command statement transfers a series of ASCII characters representing the current value of REG-3 to the DIO data portion of the IEC Bus. The series of ASCII characters is determined by the most recent FORMAT statement in the test program. If the current value of REG-3 is 13.85 and FORMAT EXP is in effect, the series of ASCII characters transferred is 0.138500E+002 where the first character preceding the 0 is a blank space that would be replaced with a minus (–) sign for negative values.
2. This second command statement will not terminate until the last position of the character string register is reached and filled (time-out applies only when DEV=BUS is specified as a sink or source in the command).

3. This third command statement terminates as soon as the last ASCII character (G) is transferred to position 14 of the character string register.
4. The result of this fourth command statement will be as expected. It is interesting to note that, invisible to the user, the system extracts the value from REG-5 in accordance with the currently established FORMAT, converts the value to ASCII characters, and then reverses the process to store the value in REG-8.
5. This fifth command statement terminates one second after the last character is transferred from the BUS (time-out).
6. A literal string *can* contain non-printing, or other special ASCII Code characters. To indicate in a string that the following three numerical digits are *not* to be taken literally, but as an 8-bit ASCII character code, the % symbol is used to precede the code. The use of the % symbol is the same as described in WRITE statements for the TEC Inc. display terminal (para. 3.4.13.2). Since a % character signifies that the immediately following 3 digits are to be taken as an ASCII character, a % cannot be sent as a literal itself. To send a %, it is necessary to send in the string a % followed by 045 (the ASCII code for %).

The character-string register, which is referenced by position lists, can be initialized with the following command:

```
CLEAR SREG;
```

This CLEAR SREG command replaces all data in the string register with ASCII NUL characters (all binary zeroes) at every position.

Character-string checking is performed with the SCOMP and STEST commands. The general forms of the command statements containing these commands are shown below:

```
optional-label: SCOMP source-1, source-2 (error-branch);
```

```
optional-label: STEST source-1, source-2 (optional-error-branch);
```

Both character-string-checking commands perform identical functions except that the STEST command also sets the fail flag (increments the FAIL register) when the optional-error-branch is taken. If an optional-error-branch specification is not included in the command statement, the system error handler will display an error message before the program proceeds to the next command.

The function performed by the SCOMP and STEST commands is a character-by-character comparison between the character strings specified as source-1 and source-2. Any source that furnishes ASCII characters can be specified for source-1 or source-2. If the character string of source-1 is not identical to the character string of source-2, the program branches as specified by the error-branch. On the other hand, the program continues with the next command statement when the character strings are a perfect match. The command terminates, passing control accordingly, when at least one of the sources is exhausted. The following are typical of character-checking command statements:

```
SCOMP 'YES', DEV=TTY [50];
```

```
SCOMP (1, 4, 6), 'NDC' [77];
```

```
SCOMP (1-5), (6-10) [113];
```

When a numerical value is specified as a source, the value is converted to a series of ASCII characters. The conversion is determined by the most recent FORMAT statement of the program. Then the character-by-character checking is performed. As an example, consider the following statement:

```
SCOMP (19-33), 1.457[9];
```

Upon execution, the numerical value specified as source-2 in the above statement is converted to an ASCII-character string in accordance with the formatting constraints currently in effect. If the most-recent format statement encountered in the test program was FORMAT 7,3, the ASCII-character contents of source-2 after conversion would appear as follows:

b	b	1	.	4	5	7
---	---	---	---	---	---	---

(b = blank or space character)

If the above series of seven characters is identical to the first seven characters (positions 19 through 25) of the character string register as specified for source-1, a perfect match is obtained and the test program continues with the next statement. Otherwise, the error-branch is taken and control passes to statement 9 of the program. Notice that positions 26 through 33 of the source-1 specification are not considered in the comparison because the exhaustion of source-2 terminates the command.

Command termination when one of the specified sources becomes exhausted is a feature that can be utilized in test programming. For example: suppose one of the devices attached to the IEC Bus is a voltmeter that, after making a measurement, can be set up as a talker to return the measured value to the system for evaluation. Furthermore, suppose that the information as returned by the device includes a four-character preamble to the measured value for indicating either a valid measurement or the occurrence of an overload. Since the preamble returned by this particular device for a normal measurement is known to be N DC, the following sequence of two statements transfers the values of valid measurements, only, to REG-19; otherwise, the program branches to statement 330:

```
SCOMP 'N DC', DEV=BUS[330];
```

```
SMOVE DEV=BUS, REG-19;
```

SBIT ON and CHECK SBIT OFF commands permit true and false checking, respectively, of selected character-bits (DIO lines). Primarily intended for checking the status byte during a serial poll, these commands are versatile and may find other uses. The functions of these commands are more readily understood from the examination of examples; hence, the following two command statements:

```
CHECK SBIT ON '%100', DEV=BUS[2001];
```

```
CHECK SBIT OFF '%100%003A', DEV=BUS[3001];
```

The first command statement shown above requests the currently established talker on the bus to present one character of information on DIO lines 1 through 8, and passes control to statement 2001 of the test program if DIO-line 7 is not a logic-1 (the logic status of the other DIO lines is not relevant). In the command statement: the check for a logic-1 is specified by the parameter ON, and DIO-line 7 is specified by the source-1 literal '%100' (the %-convention is used within the literal to indicate a 3-digit octal code, which in this case represents bit 7 true and correspondingly selects DIO-line 7).

The second command statement of the preceding group passes control to statement 3001 of the test program if the DIO lines specified by the source-1 literal are not at a logic-0 as specified by the parameter OFF. In this second example, a sequence of three characters is requested of the currently established talker to satisfy the specifications of the source-1 literal. The first specification (%100) of this command statement checks DIO-line 7 for a logic-0. The second specification (%003) checks DIO-lines 1 and 2 for logic-0s when the second character is presented by the talker. The third specification (A) is a literal character whose 7-bit ASCII representation (0100001) checks DIO-lines 6 and 1 for logic-0s when the third character is presented by the talker. However, should the talker be unable to present more than one or two characters to the BUS, only the characters presented enter into the CHECK SBIT decision and the one-second time-out then terminates the command.

As a summary of IEC Bus programming, the following program segment is an example of the sequence of command statements that might be used to perform a serial poll of 15 devices, each of which has been arbitrarily assigned a different even-numbered address:

```

20:  CHECK BUS[20];

      SET BUS UNT, UNL, SPE;

      LD 0;

      D0 15;

      SET BUS MTA(AR);

      CHECK SBIT OFF '%100', DEV=BUS [2001];

      ADD 2;

      DONE;

      WRITE DEV=TTY 'FALSE SRQ ERROR';

      BRANCH 10;

2001:  SET BUS UNT, UNL, SPD, MTA(AR);

      SMOVE DEV=BUS, (21+);

```


APPENDIX F

GR 1796 PROGRAMMING AND 1799 ANALOG PROGRAMMING

SECTION 1 — LOGIC DEVICE TESTING

F1.1 INTRODUCTION

This Appendix covers methods of preparing test programs for use on 1796 Digital/Analog Test Systems. Program preparation is achieved with a subset of the GR test language that is similar to that used for programming the other GR test systems described in this manual. Test language differences, applicable to 1796 test systems only, are contained in this Appendix. The Appendix is divided into two major sections.

Section 1 — Logic Device Testing

Section 2 — Analog/Hybrid Device Testing

This section covers Logic Device testing. It includes commands that allow functional tests to be performed at standard test speeds (for normal TTL, DTL, ECL, etc. logic family testing), and at high speeds (FAST) for dynamic functional testing of microprocessors, RAMs, ROMs, UARTs, etc.

Section 2 covers Analog testing, Hybrid testing, programming of 1796 UUT power supplies, and methods of programming the 1796/1799 pin scanner systems to connect the various source and measurement instruments and power supplies required for the testing of both analog and digital devices. Programming of unscanned devices is also covered.

F1.2 LOGIC TESTING — GENERAL

A 1796 test system tests logic circuits in either of two modes. Normal-speed testing mode is very similar to the way testing is performed on a 1792 or 1795 system. This mode has proven useful for testing such mature logic families as DTL and TTL. High-speed mode allows logic to be tested at megahertz rates; such rates permit the testing of circuits containing dynamic and/or very complex integrated circuits.

F1.3 NORMAL-SPEED TESTING

Normal-speed operation on a 1796 test system closely resembles that of a 1792-D, since system-wide signals synchronize driver and sensor operation at each test step. However, operation in this mode is fully upward-compatible with any 1792A-D or 1795 test system.

Inter-Test Time (ITT) and Inter-Strobe Time are also quite similar to those for a 1792-D, and vary with the same parameters. The parameter values for a 1796 are given in Table F-1.

F1.4 HIGH-SPEED TESTING

All features of the high-speed testing option are controlled by statements in the test language, as described in the following paragraphs. Paragraph F1.4.1 describes hardware operation as it affects test program execution, and the interaction between software and hardware is briefly explained in F1.4.2 and F1.4.3; the latter paragraph also shows how test timing can be computed.

Test statements which are allowed inside a high-speed program segment are described in detail in paragraph F1.4.4. General considerations and constraints on the construction of high-speed test programs are given in paragraph F1.4.5. Finally, paragraph F1.4.6 shows an example of a high-speed test program.

F1.4.1 Hardware Overview

Installation of the high-speed driver/sensor system option in a 1796 test system provides the capability for logic testing at test rates up to 1.8 MHz. The option consists of two parts: the high-speed controller, and high-speed memory.

Table F-1 (1,2)
1796 NORMAL-SPEED TEST TIMING

	SET LGC (No Label)	SET LGC (With Label)	TEST LGC (No Label)	TEST LGC (With Label)
Time to Load Drivers, μsec	8b	8b	8b	8b
Default IST, μsec	none	none	3.2	3.2
Compare Time, μsec	none	none	12 + 11B	12 + 11B
Software Overhead and Label Processing Time, μsec	33	227	43	238
Total ITT in μsec =	33 + 8b	227 + 8b	58.2 + 8b + 11B	253.2 + 8b + 11B

At any test group of contiguous boards, beginning with the boards containing the lowest-numbered driver that changes state and going through the highest-numbered driver that changes state, is loaded with state information. The number of boards in this group is computed as:

$$b = \left\lceil \frac{\text{HIDP}}{12} \right\rceil - \left\lceil \frac{\text{LODP}}{12} \right\rceil + 1 ;$$

HIDP = highest driver pin that changes

LODP= lowest driver pin that changes

$\left\lceil \right\rceil$ = enclosed result rounded to next higher integer.

(1) B = total number of driver/sensor boards in the system

(2) b = number of driver/sensor boards on which driver states change at this test step.

The high-speed controller is, in effect, a special-purpose digital processor. It executes a set of instructions which are closely tailored to the basic functions of logic testing, and which therefore are quite similar to logic statements in the test language. Certain test language statements, such as SET LGC and TEST LGC, correspond to commands which are directly executable by the controller. The controller also can execute DO loops at high speed. Another useful controller function is its ability to synchronize a test sequence with a UUT signal, by waiting for the signal to reach a specified state before allowing execution to continue.

Providing driver/sensor pins with the optional high-speed memory allows them to be used for high-speed testing. The memory can store one bit of state information per driver, for each test step up to a maximum of 4,095, and can record the results of each test. A single memory board provides storage for 24 driver/sensors. High-speed testing can be performed on only those driver/sensors with associated high-speed memory.

F1.4.2 STRUCTURE OF FAST BLOCKS

Software control of high-speed test sequences is established by defining a "FAST block". A fast block is a sequence of test statements which are to be executed by the high-speed controller, in a burst, when they are encountered in a test program. The definition of a FAST block consists of a BEGIN FAST instruction immediately before the first instruction of the block, and a FINISH FAST instruction immediately after the last instruction of the block.

Instructions within a FAST block must be "legal" FAST instructions, i.e. those which are executable by the high-speed controller. Legal FAST instructions are listed in Table F-2. An attempt to insert any other type of instruction in a FAST block will result in a Translator error.

Table F-2
FAST INSTRUCTIONS

Command	Required Arguments	Optional
SET LGC	— none —	pin list(s)
TEST LGC	— none —	pin list(s) error branch
REASSIGN LGC	pin list(s) —	— none —
IGNORE LGC	pin list(s) —	— none —
BEGIN FAST	pin list(s)	IST specification ITT specification FAST option list(s)
FINISH FAST	— none —	user clock
SET FAST	— none —	FAST option lists
DO	loop count	— none —
DONE	— none —	— none —
NOP	— none —	— none —
BEGIN RECORD	— none —	— none —
FINISH RECORD	— none —	— none —
BLOCK	— none —	— none —

F1.4.3 FAST Block Operation and Timing

Software execution of a FAST block is performed in three sequential modes: Load mode, Burst mode, and Result mode.

Load Mode

Load mode is entered from normal speed operation when a BEGIN FAST command is encountered. In this mode all test patterns and other parameters for the FAST block are loaded into the high-speed controller and memories. There is no activity at the UUT shelf during Load mode. The duration of Load mode is proportional to the amount of data loaded into the high-speed memory and controller, number of FAST test steps, pins, DO loops, etc.

Burst Mode

After loading is completed the Burst mode is entered. At this time the computer passes control to the high-speed controller, which begins to execute the test sequence defined by the FAST block. The test rate during Burst mode is determined by the IST and ITT parameters loaded during the previous mode.

The duration of Burst mode is equal to the product of ITT and the number of SET/TEST statements in the FAST block. A statement in a DO loop is counted as occurring once for each time the loop is iterated.

Result Mode

When the controller finishes executing the last statement in the FAST block, its operation ceases. Control is passed back to the computer, which enters Result mode. The results recorded for each test statement in the FAST block are fetched from the high-speed memories and analyzed in the order in which they were executed during Burst mode. Errors, which are defined as differences between expected signal outputs and outputs actually recorded during the test sequence, are discovered at this time. Error branches and error halts associated with statements in the FAST block are performed when the errors are actually discovered, during Result mode.

If Result mode processing is completed with no error actions outstanding, the computer returns to normal-speed execution mode. Further considerations regarding the use of FAST blocks are given in section F1.4.5.

Figure F-1 shows the timing that results when a FAST block is encountered during normal-speed testing.

F1.4.4 Description of FAST Block Commands

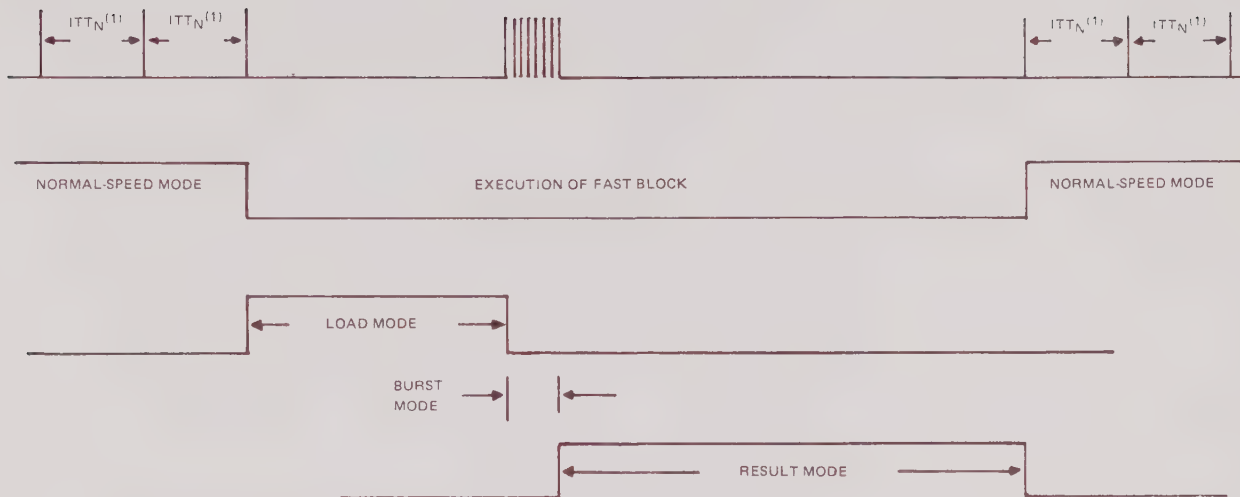
Logic commands in FAST Blocks

Operation of the SET, TEST, REASSIGN, and IGNORE LGC commands in high-speed mode differs only slightly from their operation in normal-speed mode.

```
SET LGC IH(9, 3) IL(12, 2) OH(16, 13) OL(5, 8);
```

The format and operation of the SET LGC command are identical for FAST blocks and normal test program segments, with one exception: ITT and IST specifications are not allowed in a SET LGC statement within a FAST block, since these values cannot be altered during execution of a FAST block.

```
TEST LGC IH(9, 3) IL(12, 2) OH(13, 16) OL(5, 8) [4109];
```



(1) ITT_N = NORMAL-SPEED INTER-TEST TIME

Figure F-1. FAST Block Operation and Timing

Operation of the TEST LGC command in a FAST block differs from that in a normal program segment in two respects. First, ITT and IST specifications are not allowed in a TEST LGC command within a FAST block, because the high-speed controller cannot alter these quantities during execution of a FAST burst. Second, the results of the TEST are not known until Result mode processing, which is performed after the high-speed operation is complete; if an error branch is specified it will be taken in Result mode, rather than during the high-speed burst.

IGNORE LGC (20, 21, 22, 34);

The format and execution of the IGNORE LGC command are identical for FAST blocks and normal-speed test sequences. It should be remembered that IGNORE refers to error-checking, and so only applies to Result mode.

REASSIGN LGC I (5, 12) 0(4, 6);

THE REASSIGN LGC statement has the same function in FAST blocks as in normal-speed operation; that is, it enables or disables drivers. However, in Burst mode driver assignment is performed by the high-speed controller. The high-speed memories store up to 15 distinct assignment patterns, one of which must be the pattern derived from the last ASSIGN LGC statement encountered before the FAST block. Any number of REASSIGN statements is allowed so long as no more than 15 distinct patterns are generated; exceeding this limit **will** result in an Interpreter error.

Execution of a REASSIGN LGC statement within a FAST block with an IST of less than 650 nsec. will result in DIAGNOSIS Mode errors, and should be avoided.

BEGIN FAST Commands

BEGIN FAST (1-7, 9, 12, 15-48) ITT=550N HI (WAIT-7);

The BEGIN FAST statement denotes that the sequence beginning with the next SET or TEST LGC statement is to be executed in Burst mode. It also determines those pins which will be active during the burst. Any pin not in this active pin list will be considered "normal", and its state can **not** be changed during this FAST block. However, normal pins may be assigned as drivers and used as static stimuli during a FAST block. Note that a pin may be "normal" even if it has high-speed memory. Normal pins are ignored during FAST block operation.

In the example given pins 1 through 7, 9, 12, and 15 through 48 can change state during the block following the BEGIN FAST statement. Pins 8, 10, 11, 13, and 14, although present in the system, will be treated as normal pins during this FAST block; an attempt to change the state of any of the normal pins will result in an Interpreter error.

A BEGIN FAST statement may optionally specify Inter-Test Time (ITT) and Inter-Strobe Time (IST). ITT may be specified with a resolution of 25 nanoseconds within the range 550 nanoseconds to 102.4 microseconds. The resolution of IST is also 25 nanoseconds; its range is from 150 nanoseconds to a value 100 nanoseconds less than the value of ITT. Alternatively, specifying IST=Z causes the controller to issue IST and ITT simultaneously.

HI and LO option lists may also be specified within a BEGIN FAST statement, as described in the section under SET FAST (see section F1.3.4.4).

FINISH FAST Commands

FINISH FAST CLK;

A FINISH FAST statement identifies the last previous SET or TEST LGC statement as the last statement in a FAST block. Inclusion of the optional CLK argument enables the User Clock, user Driver Strobe, and User Sensor Strobe signals at the UUT shelf to continue to run after the high-speed controller has halted. This allows a UUT employing dynamic circuitry to be kept "alive" and in a known state until execution of a subsequent FAST block can begin.

SET FAST Commands

SET FAST HI(WAIT-7) LO(USER-1 EXIT-48);

FAST options which apply during Burst mode may be turned on or off by the SET FAST command. Arguments in a SET FAST statement are HI or LO option lists, containing FAST option specifications and their related pin numbers. The three FAST option types are:

WAIT-n synchronizes the high-speed processor operation with a UUT signal. If WAIT is specified for a TEST or SET statement, execution of following TEST or SET statements will not begin until the signal reaches the stipulated state. A WAIT condition applies to all following test statements until a later WAIT specification removes the condition or specifies a new one. To remove the WAIT, use a SET FAST statement with **no** WAIT condition specified. The pin number refers to one of seven "wait" signals from the UUT shelf to the controller which can be selected by this option. Execution continues when the selected signal reaches the state specified, either HI or LO. If the condition is not met within 30 ms, an error may result, due to the dynamic refresh requirements of the high-speed memory. Thus, in the example:

```
BEGIN FAST (1-12);  
SET FAST HI (WAIT-7);  
5:  TEST LGC IH (1-6)IL (7-12);  
10: TEST LGC IL (1-6)IH (7-12);
```

WAIT-7 is set up with statement 5: and statement 10: will not execute until signal 7 goes high. The WAIT-n signals are defined as SYNC 1-7 on the shelf (J29).

In the example of Figure F-2, Driver Strokes for steps 2 and 4 are not issued until the WAIT-4 signal goes high, some time after ITT. Test step 3 begins immediately after ITT, since WAIT-4 is still high at that time. Step 5 begins after ITT, even though WAIT-4 is low, since the SET FAST statement preceding step 4 removes the WAIT condition.

USER-0 and USER-1 are TTL signals which are brought out from the high-speed controller to the UUT shelf. They may be specified to LO or HI at any step in a FAST block. Once set, they remain in that state until changed by a subsequent SET FAST or BEGIN FAST command. Until they are set, they are both HI. The given example sets the USER-1 signal to a LO beginning with the next test step.

The EXIT-n option allows an exit from a DO loop when a specified sensor goes into a certain state. The exit is taken at completion of the pass through the loop on which the sensor is found in the specified state. An EXIT option may refer to any sensor which was included in the active pin list in the last BEGIN FAST statement. Only one EXIT condition is allowed for each loop; if a loop contains more than one EXIT condition, only the last one specified before the DONE statement has any effect. In the example shown an exit will occur at completion of the DO loop as soon as sensor 48 has gone LO.

When more than one of the Wait, User, or Exit options is included in a SET FAST statement a space and NOT a comma is used as a separator.

Each SET FAST command affects the state of all FAST options. Those mentioned in the statement are set to the specified states; if not specified, they are turned off. Once set, they remain in the specified state until changed by a subsequent SET FAST command.

FAST options in SET FAST commands are actually executed, during Burst mode, on the test step associated with the next subsequent SET or TEST command. An EXIT command which is the last statement in a DO loop (just prior to DONE) will cause the loop exit as described previously. However, a USER or WAIT commands should not be used as the last statement in a loop.

The use of FAST options and option lists by the BEGIN FAST command is identical with that described for the SET FAST command.

```

BEGIN FAST (44) IST=1U ITT=2U HI(WAIT-4);
1: IH(44);
2: IL(44);
3: IH(44);
   SET FAST;
4: IL(44);
5: IH(44);

```

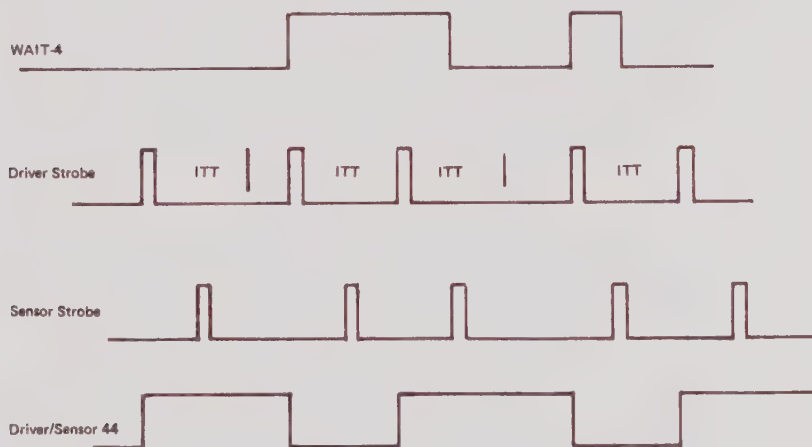


Figure F-2. Example of WAIT Operation.

DO and DONE Command Pairs

```
DO 64;  
$IH(191);  
$IL(191);  
DONE;
```

The structure of DO loops is basically the same in both high- and normal-speed testing. The example shown will simply produce 64 positive-going pulses at UUT pin 191. DO loops within a FAST block are executed by the high-speed controller, and so certain considerations apply to this case which are **not** true for normal-speed testing.

Up to eight separate DO loops are allowed within each FAST block. DO loops in a FAST block may not be nested. The loop count, which is set by the initial DO command, can be any positive number between one and eight million. In fast sections, a loop count of zero will cause the loop to repeat 2^{24} times, unless it terminates for some other reason.

The other way a DO loop can terminate is by encountering a SET FAST command with an EXIT-n argument. This will cause the loop to exit on the DONE statement after the specified sensor state is reached. Use of the ESCAPE command is not allowed in FAST blocks.

There must be at least one SET or TEST LGC statement between the end of a DO loop and the start of the next loop in a FAST block. Inside the loop only SET LGC and SET FAST commands are allowed; the loop must contain at least two SET commands.

F1.4.5 Programming Considerations

The following general considerations apply to FAST blocks;

A FAST block can contain a maximum of 4,095 test patterns. (A test pattern is created by each SET or TEST LGC statement.) SET FAST, BEGIN and FINISH FAST, DO and DONE, REASSIGN, and IGNORE statements are merged with SET or TEST statements, and consequently require no space in high-speed memory.

There must be **at least one** SET or TEST LGC statement between each BEGIN FAST statement and the last previous ASSIGN LGC statement.

A FAST block may **not** be executed by issuing a CALL instruction with an active pin list.

When programming driver logic levels, VIH must always be at least 0.3 volts greater than VIL. An Interpreter error will result if this relationship is violated.

When operating in systems equipped with CAPS, FAST blocks must be entirely inside or entirely outside CAPS sections (as defined in section 3.2.5.) The Diagnostic Resolution Module and FINDS software options, if installed, are also restricted in a CAPS section containing a FAST block: the guided clip and microvoltmeter tests are bypassed in that case, and only short-circuit tests are performed. The full fault resolution capabilities of the DRM can be brought to bear for normal-speed testing by placing FAST blocks in separate CAPS sections.

F1.4.6 FAST Programming Example

The network of Figure F-3 will be used as an example of the uses of high-speed operation on the 1796. It is taken from a board used as a circulating buffer memory in the TEC data terminal on the 1796. The memory is seven bits wide by 2050 bits long; the Figure shows only bits one and seven. IC14 and 15, 24 and 25, etc. through ICs 74 and 75 are 1024 bit PMOS dynamic shift registers; they require clock drivers CD1 and 2. The remaining circuitry shown is standard TTL logic.

A program to test the board is shown in Figure F-4. It first defines two MACROs to pulse the clock (D/S 36): one, 'SCLOCK', uses SET LGC statements, and is to be used in DO loops. The 'TCLOCK' MACRO uses TEST LGC statements, and will be used wherever it is desired to record sensor states.

The first CAPS section, which is named 'NORMAL', tests the clock drivers and associated normal-speed logic. Since it contains no FAST blocks, the full capabilities of FINDS and guided clip software can be used in this section if an error is detected. (In actual practice all of the non-dynamic circuitry would be tested in this section, to maximize the fault resolution on all areas of the board. Additional tests are omitted here for clarity.) Note that logic level set B is assigned to sensors 110 and 111, which connect to test points at the outputs of the clock drivers.

The second CAPS section, 'FAST', which begins on statement number 8, tests the dynamic memory and associated logic at its maximum clock rate. Statement 10 separates the ASSIGN and BEGIN FAST commands, as required, as well as initializing drivers 25 and 36. High-speed testing then begins with ITT set to 675 nanoseconds, which corresponds to the maximum clock rate specified for the dynamic shift registers, 1.5 MHz.


```
.DEFINE 'TCLOCK';
    IH(36);
    IL(36);
.ENDM;

.DEFINE 'SCLOCK';
    $IH(36);
    $IL(36);
.ENDM;

        BEGIN CAPS 'NORMAL';

1:          ASSIGN LGC IA(6, 25, 36, 45, 47, 49, 51, 53, 55, 57) VIHA=4.0
VILA=0.4
                OA(2, 9, 10, 42, 62, 71, 46, 48, 50, 52, 54, 56, 58)
VOHA=3.5  VOLA=1.0
                OB(110, 111) VOHB=4.0 VOLB=-10.0;

2:          SET PS-1 V=5 I=2;
            SET PS-4 V=-5 I=1;
            SET PS-5 V=-12 I=1;

4:          IH(6);
5:          IH(36);
6:          IL(36);
7:          IH(36);

        FINISH CAPS;

8:          BEGIN CAPS 'FAST';
9:          ASSIGN LGC IA(6, 25, 36, 45, 47, 49, 51, 53, 55, 57) VIHA=4.0
VILA=0.4
                OA(2, 9, 10, 42, 62, 71, 46, 48, 50, 52, 54, 56, 58)
VOHA=3.5  VOLA=1.0
                OB(110, 111) VOHB=4.0 VOLB=-10.0;

10:         IH(25) IL(36);/          ENABLE CLOCK      /
11:         BEGIN FAST (25, 26, 36, 45-58, 110-111) IST=525N ITT=675N;

;
12:         DO 2050;
            .'SCLOCK';
13:         DONE;
14:         OL(46, 48, 50, 52, 54, 56, 58) [31];

15:         IH(45, 47, 49, 51, 53, 55, 57);/ INPUT BITS=1  /
```

Figure F-4. Programming Example (sheet 1 of 2).

```

16:          DO 2050;
17:             SET FAST HI(EXIT-46);
             . 'SCLOCK';
19:          DONE;

21:          OH(46, 48, 50, 52, 54, 56, 58) [31];

22:          IL(45, 47, 49, 51, 53, 55, 57);/ SET DATA INPUT ALL LOW  /

23:          .REPEAT 4;
          .SHIFTO (45, 47, 49, 51, 53, 55, 57);
          . 'TCLOCK';
          .SHIFT1 (45, 47, 49, 51, 53, 55, 57);
          . 'TCLOCK';
          .LOOP;

24:          DO 1017;
          . 'SCLOCK';
          DONE;

25:          .REPEAT 7;
          . 'TCLOCK';
          .LOOP;

26:          DO 1017;
          . 'SCLOCK';
          DONE;

          IL(25);/          ENABLE CIRCULATION  /

          .REPEAT 14;
          . 'TCLOCK';
          .LOOP;

28:          FINISH FAST;

29:          FINISH CAPS;

30:          BRANCH 32;

31:          SMOVE '
BOARD FAILED TO INITIALIZE !
', DEV=TEC;

32:          NOP;
          END;

```

Figure F-4. Programming Example (sheet 2 of 2).

The first testing step is performed by the DO loop beginning at label 12. With all data input bits low the memories are clocked 2050 times, which should make the data output bits go low. A test for this condition is then performed, and an error displayed if it is not met (and if CAPS is not being used.)

Next, the data input bits are set high and the DO loop of statements 16 through 19 is executed. The DO loop clocks the register 2050 times, after which all of the data output bits should be high. As an added precaution, sensor 46 is monitored in case it goes high prematurely. A test is then made to see whether any output bits have remained low, as would be the case if a data node was stuck in that state.

A sequence of eight patterns of alternating ONEs and ZEROS is next loaded into the shift register, and clocked by the DO loop beginning at label 24 until it is about to appear at the outputs of the first row of dynamic memory chips. The .REPEAT MACRO at label 25 allows the nodes connecting the pairs of dynamic chips to be tested, and clocks the pattern until it is shifted into the second row of chips. The DO loop then shifts it through to the outputs, after which the final .REPEAT MACRO tests the nodal values there.

Both the FAST block and the second CAPS section terminate at this point. The rest of the program consists of statements for displaying an error message.

SECTION 2

ANALOG/HYBRID DEVICE TESTING

F2.1 INTRODUCTION

This section of Appendix F contains an overview of the 1796 and 1799 pin scanner systems used to multiplex source and measure instruments onto scanned pins at the analog and digital shelves (and/or at the analog probes). The 1796 scanner system is described in para. F2.2.1 while the 1799 scanner system is described in para. F2.2.1A. An overview of the separate UPS Device Power Supply scanner system is also provided. Methods of programming the two scanner systems to perform a variety of tasks (such as the set-up and interconnection of the proper source and/or measurement instruments or UPS power supplies for test purposes) are also described. Finally, methods of programming any non multiplexed devices (such as the Time Measurement Unit, relay drivers, etc.) are covered.

F2.2 PIN SCANNER SYSTEM

F2.2.1 1796 Overview

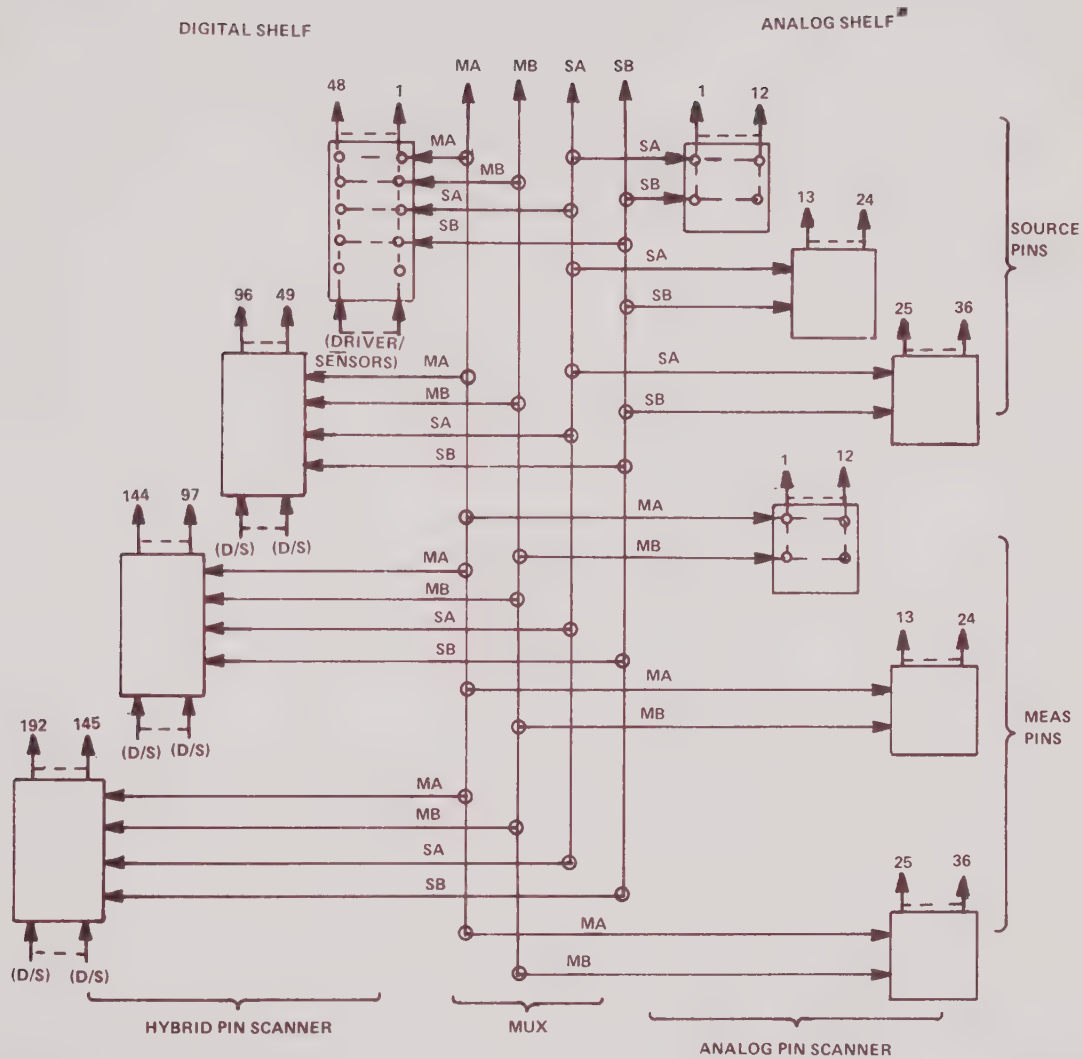
The 1796 hybrid/analog pin scanner system interconnects a series of measurement instruments, and source instruments with selected UUT shelf pins under test program control (see Figure F-5). Source instruments may be connected to two source channels (SA, SB), while measurement instruments may be connected to two measure channels (MA, MB) through activation of program controlled multiplexed (MUX) mercury-wetted cross point relays. The relay matrix serves to connect source instruments or measurement instruments to the designated source or measure channel when directed by the test program. The source and measure channels are connected, in turn, to an analog pin scanner, optional hybrid pin scanner (that also interconnects Driver/Sensors), or analog probe scanner. Each scanner consists of dry cross point relays that actually interconnect the specified source or measure channel to selected shelf pins for UUT testing.

The first 192 pins on the digital shelf may be used as hybrid pins to interconnect driver/sensors, or source or measure channels (or both) to the shelf pins through the optional hybrid pin scanner. In a similar fashion, the analog pin scanner connects multiplexed source or measure channels (or both) to analog pins on the analog shelf. The analog probe scanner also connects the source or measurement channels to any one of a group of analog probes that allow testing at points internal to the UUT. Connections are also provided on the analog shelf, at the rear of the analog scanner box, and at the rear of the analog probe scanner box for attachment of external source and measurement instruments to selected channels.

The power supply pin scanner is a **separate** system consisting of mercury-wetted relays that interconnect UPS device power supplies with UUT pins on the analog shelf. Only the UPS power supplies can be scanned. Up to five UPS power supplies may be scanned to pins on the analog shelf under test program control. Both High (VHI, IHI) and Low (VLO, ILO) voltage sense and current leads are scanned to the shelf pins where they may be Kelvin-connected in the device adaptor at the shelf. The non-Kelvin-connected HI and LO outputs of the UPS power supply installed as PS-2 are also supplied to the analog probe scanner where they can be multiplexed onto either probe source channel (SA or SB). The power supply scanner system is not programmed in the same manner as the hybrid and analog scanner system described in the following paragraphs. Methods of programming the power supply scanners are described in Para. F2.4 and 3.2.7.3.

The 1796 Hybrid/Analog scanner system is designed to minimize cross-talk by disconnecting all digital and analog shelf pin groups that are not being used as each test program SET, MEAS, or TEST statement with an AT construct is executed in sequence. Analog shelf scanner pins are assigned in groups of twelve source pins and twelve measure pins while digital shelf hybrid pins are assigned in groups of forty-eight pins (see Figure F-6). When a new test program scanner AT construct does not use any of the pins in a previous AT construct group, that group is disconnected at the MUX thus eliminating possible wire stubs that are no longer required for test purposes, yet might cause impedance and transmission problems. SET, MEAS, or TEST statements that do not contain AT constructs allow MUX scanner pins to remain in the condition they were in previously.

Pin groups that have shelf pins which are included in **both** old and new scanner AT constructs are not normally disconnected from the MUX so that steady (uninterrupted) states can be maintained as each statement is processed. Connections to the analog or hybrid shelf pins that are to utilize the source or measure channels should therefore be made within the **same** group of 12 (or 48) pins whenever possible, in order to take maximum advantage of the 1796



*Supplied with 1796 Test Systems only.

Figure F-6. Hybrid and Analog Pin Scanner Shelf Connections.

Hybrid/Analog group switching scheme. This can be accomplished by suitable pin selection during construction of the UUT device adaptor and initial program preparation.

The following instruments may be connected to the source or measure channels (SA, SB, MA, MB) shown in Figure F-5 through use of test program SET, MEAS, and TEST statements:

1. Up to two programmable pulse generators (PGEN-1, PGEN-2) that produce pulse trains or bursts of pulses for use as UUT stimulus signals. A single pulse generator (PGEN-1) is optionally available. A second pulse generator (PGEN-2) may be added to some special test systems as a custom product.
2. Up to two programmable function generators may be used as ac sources (ACS-1, ACS-2). These units produce sine, square, and triangular waveshapes over a wide range of amplitude and frequency. Both PGEN and ACS source outputs may be directed to **unscanned** pins at J24 or J28 on the analog shelf (in addition to the scanned pins) whenever the keyword SHELF is included in SET, MEAS, or TEST statements.
3. A programmable frequency measurement unit (FMU) that can measure frequency, frequency ratio, time intervals, and also count events over a wide range.
4. A precision 16-bit multimeter for high accuracy measurements. The basic unit measures both dc volts (DCV) and dc current (DCI). Optional additions allow ac voltage (ACV), resistance (RMU), and a high compliance current source for non-linear network measurements (DCVIF) to be added (see Figure F-7). The precision multimeter may be externally triggered (EXTS) for track and hold operations (where an external event is allowed to initiate a measurement) and has the following additional features:
 1. A low pass filter (LPF) that filters out ac components not necessarily power line related, or high frequency noise that may be superimposed on the signal at power line frequencies. It has a noise rejection of greater than 20dB at power line frequencies.
 2. A power line frequency integrate mode (INTG) that removes noise due to power line pick-up. It provides greater than 40dB rejection of power line related noise.
 3. A peak detector mode (nPEAK) that enables the maximum positive (PPEAK) or negative (NPEAK) peaks of waveforms to be detected. For this measurement, peaks are considered to be the most positive (or most negative) full-scale position.
 4. A direct mode (not shown) that enables direct output of measured results without using the nPEAK, INTG, or LPF modes. Direct outputs are not filtered or integrated in any manner.

The analog probe scanner interconnects the source and measure channels to two jacks labelled PROBE1 and PROBE2 at the front of the probe scanner box (see Figure F-8). A general purpose (TYPE=1) probe and an analog diagnostics probe (TYPE=2) are currently available. Either of these probes can be connected to the PROBE1 or PROBE2 jacks and interconnected to the source and measure channels through test program SET PROBE statements. A spare instrument may also be connected at J3 on the rear of the analog probe scanner box and interconnected to the probe channels. The High and Low non-Kelvin connected outputs of UPS power supply PS-2 may also be connected to probe source channels A and B through the analog probe scanner box.

Both analog probes are hand-held devices that enable users to probe circuit points that are not normally accessible at UUT edge connector pins. Both probes contain a set of four relays that may be operated independently by test program SET PROBE statements to interconnect the correct source or measure channels through to the probe tip. The TYPE=1 general purpose probe may be used to apply direct source and direct measure signals to the UUT, or ac coupled measure signals, or make single point impedance measurements, etc. The TYPE=2 probe is designed primarily for use with optional GR analog diagnostics software and is described in the analog diagnostics manual supplied with that software. Methods of programming the multiplexed and non-multiplexed source and measurement instruments, and device power supplies are described in the following paragraphs.

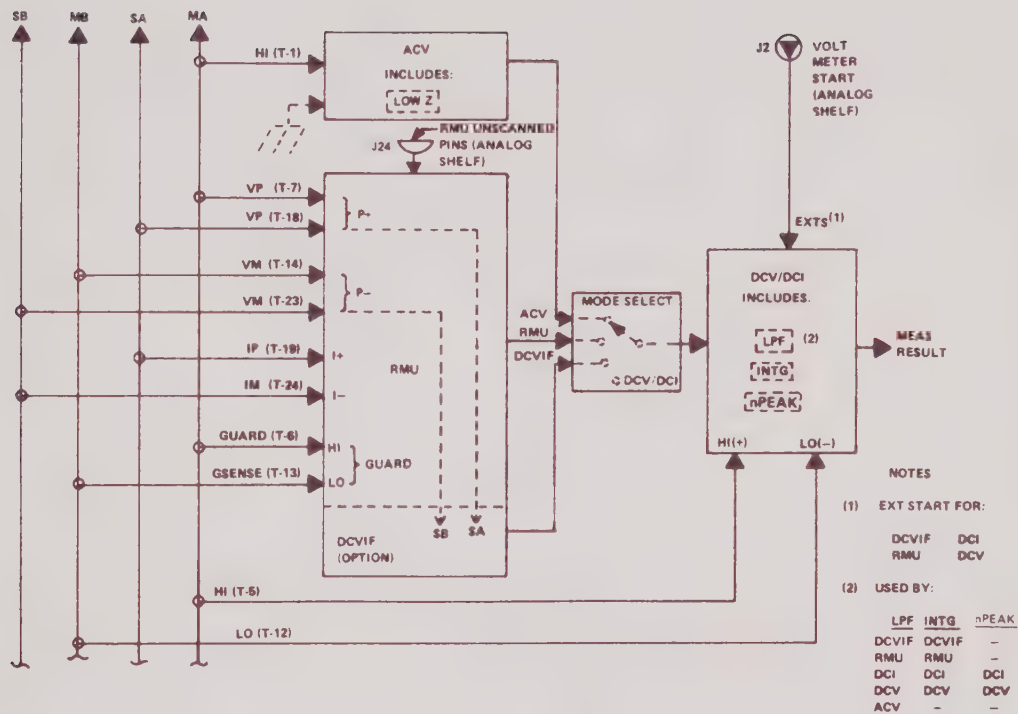


Figure F-7. Precision Multimeter Scanner Connections (1796).

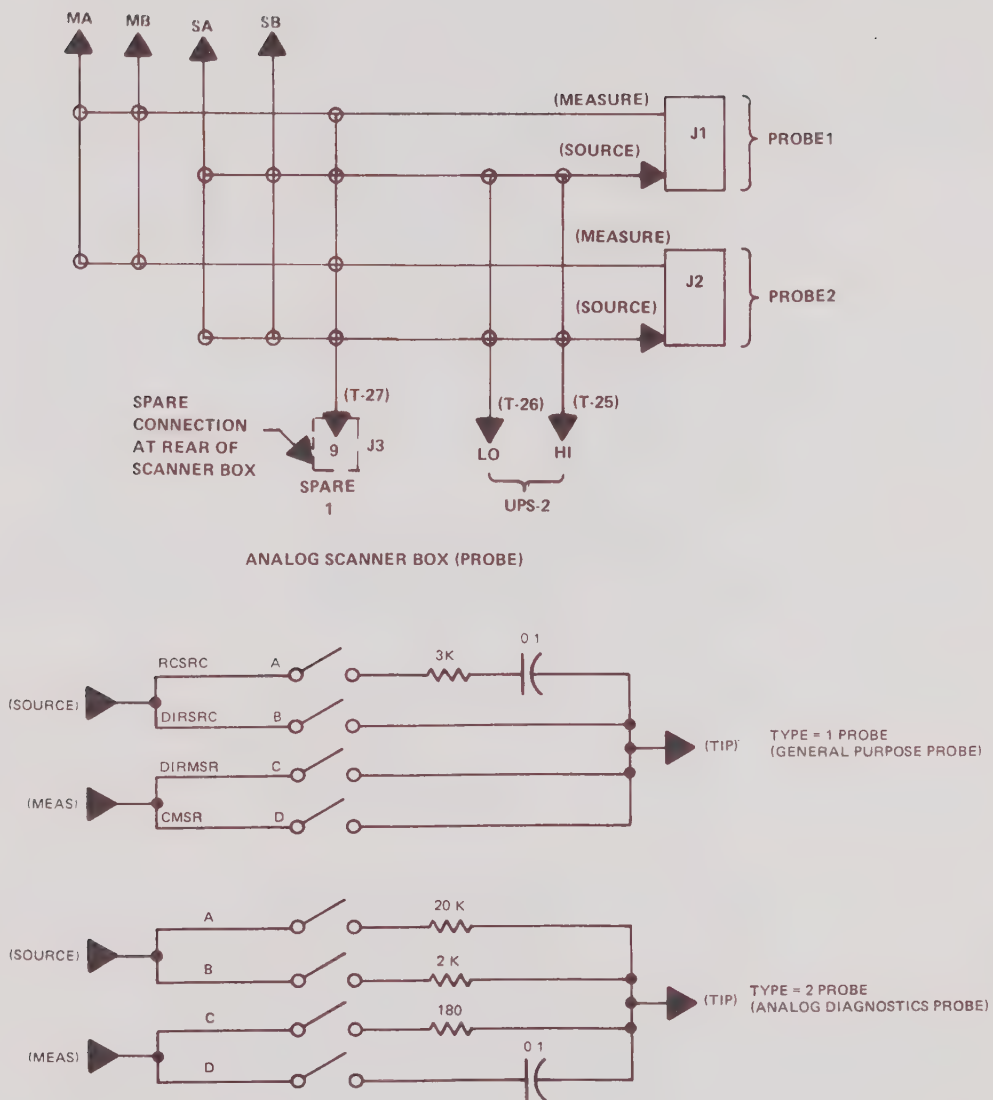


Figure F-8. Analog Probe Scanner Connections.

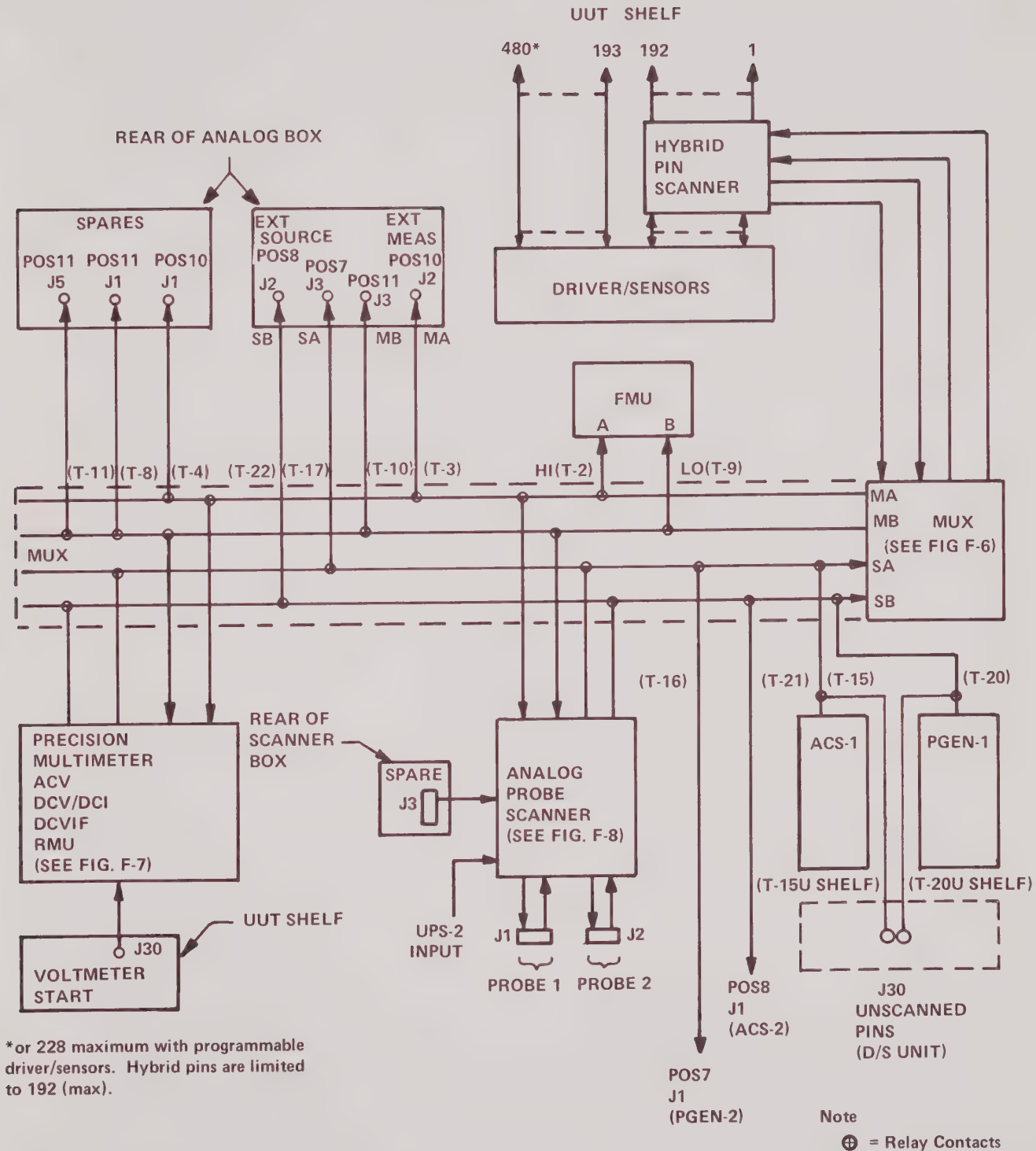


Figure F-8A. 1799 Pin Scanner Connections.

F2.2.1A 1799 Overview

The 1799 hybrid pin scanner system interconnects a series of measurement instruments, and source instruments with selected UUT shelf pins under program control (see Figure F-8A). Source instruments may be connected to two source channels (SA, SB), while measurement instruments may be connected to two measure channels (MA, MB) through activation of program controlled multiplexed (MUX) mercury-wetted cross point relays. The relay matrix serves to connect source instruments or measure instruments to the designated source or measure channels when directed by the test program. The source and measure channels are connected in turn to a hybrid pin scanner (that also interconnects Drivers/Sensors), or analog probe scanner. Each scanner consists of dry cross point relays that actually interconnect the specified source or measure channel to selected shelf pins for UUT testing.

The first 192 pins on the digital shelf can be used as hybrid pins to interconnect driver/sensors, or source and measure channels (or both) to the shelf. The analog probe scanner connects the source or measure channels to any one of a group of analog probes that allow testing at points internal to the UUT. Connections are also provided at the rear of the analog measurement box for attachment of external source and measure instruments to selected channels.

The 1799 Hybrid scanner system (like the 1796) is designed to minimize cross-talk by disconnecting all digital and hybrid shelf pin **groups** that are **not** being used as each test program SET, MEAS, or TEST statement with an AT construct is executed in sequence. Hybrid pins are assigned in groups of forty-eight pins (see Figure F-8A). When a new test program scanner AT construct does not use **any** of the pins in a previous AT construct group, that group is disconnected at the MUX thus eliminating possible wire stubs that are no longer required for test purposes, yet might cause impedance and transmission problems. SET, MEAS, or TEST statements that do **not** contain AT constructs allow MUX scanner pins to remain in the condition they were in previously.

Pin groups that have shelf pins which are included in **both** old and new scanner AT constructs are not normally disconnected from the MUX so that steady (uninterrupted) states can be maintained as each statement is processed. Connections to shelf pins that are to utilize the source or measure channels should therefore be made within the **same** group of forty-eight pins whenever possible, in order to take maximum advantage of the 1799 Hybrid group switching scheme. This can be accomplished by suitable pin selection during construction of the UUT device adaptor and initial program preparation.

The following instruments may be connected to the source or measure channels (SA, SB, MA, MB) shown in Figure F-8A through use of test program SET, MEAS, and TEST statements:

1. Up to two programmable pulse generators (PGEN-1, PGEN-2). Pulse generators produce pulse trains or bursts of pulses for use as UUT stimulus signals.
2. Up to two programmable function generators may be used as ac sources (ACS-1, ACS-2). These units produce sine, square, and triangular waveshapes over a wide range of amplitude and frequency. Both PGEN-1 and ACS-1 source outputs may be directed to unscanned pins at J30 on the Driver/Sensor unit (in addition to the scanned pins) whenever the keyword SHELF is included in SET, MEAS, or TEST statements.
Driver/Sensor unit (in addition to the scanned pins) whenever the keyword SHELF is included in SET, MEAS, or TEST statements.
3. A programmable frequency measurement unit (FMU) that can measure frequency, frequency ratio, time interval, and also count events over a wide range.
4. A precision 16-bit multimeter for high accuracy measurements. The multimeter is identical to the one used in 1796 Test Systems and is described in para. F2.2.1.
5. An analog probe scanner system that enables users to probe circuit points that are not normally accessible at the UUT edge connector pins. The probe scanner is identical to the one used in 1796 Test Systems and is described in para. F2.2.1.
6. Power Supply Scanning is **not** available in 1799 Test Systems. The device power supply configurations are described in Para. D1.4 of Appendix D.

F2.2.2 Overall Hybrid/Analog Scanner Language

The general format of a Hybrid/Analog scanner statement is:

For 1796 Analog Pins				For 1796/1799 Hybrid Pins		
MEAS	DCV	AT(MA@1:MB@2);	or	MEAS	DCV	AT (MA@C=1:MB@C=2);
<u>Oper- ation</u>	<u>Device</u>	<u>Device Specification</u>		<u>Oper- ation</u>	<u>Device</u>	<u>Device Specification</u>

Where:

MEAS	Defines the operation to be performed (in this case MEAS). Either SET MEAS or TEST operations may be specified.
DCV	Defines the device (in this case DCV). Any of the devices listed in Tables F-3 (for a 1796) and F-3A (for a 1799) may be specified (ACV, FMU, RMU, etc).
AT	Means attach the device specified to the channel, scanner shelf pins, etc. listed in the device specification. The AT construct must be included in SET, MEAS, or TEST statements if MUX channels and/or scanner pins are to change state. If an AT construct is not included, all connections will remain in their previous state.
(MA@1:MB@2) or (MA@C=1:MB@C=2)	This particular device specification lists the channels that will be connected to UUT MEAS pins on the 1796 analog shelf (connect the MA channel to MEAS pin 1 and the MB channel to MEAS pin 2), or through the 1796/1799 hybrid pins (connect the MA channel to hybrid pin 1 and the MB channel to hybrid pin 2). The @ symbol must separate the terminal/channel/pin list from any pin list that follows. In addition, a colon (:) must separate each terminal/channel/pin list from the next one to be specified (if any). The C= token assigns pins to the hybrid pin scanner.
	The semi-colon (;) is a line terminator that must end all statements.

NOTE

Many programming examples in this Appendix list pin assignments for the analog shelf only. All pin assignments for hybrid pins **must** include C= or H= tokens in the AT construct. For example,

MEAS DCV AT (1:2);

assigns the DCV to pins on the analog shelf while

MEAS DCV AT (C=1:C=2);

assigns the DCV to the hybrid pins.

Device specifications may include either implicit or explicit commands, but not both. Implicit commands are abbreviated commands that cause the translator to make automatic assignments as in the following example:

For Analog Pins

MEAS DCV AT(1:2);

For Hybrid Pins

MEAS DCV AT (C=1:C=2);

This example performs a function which is identical to the longer explicit AT (MA@1:MB@2) examples listed above. However, since they are abbreviated, it may be more difficult to understand if added comments are not also included in the test program. Either the channel names or terminal names/mnemonics listed in Table F-3 and F-3A may be omitted from an AT construct and the translator will make automatic assignments of the others. Terminal name and mnemonic assignments allow explicit direction of a device to the multiplexer. For example,

MEAS DCV AT(HI MA@1:LO MB@2);

is identical to

MEAS DCV AT(1:2);

However, it explicitly shows that the High (1) DCV terminal is connected via the MA channel to MEAS pin 1 on the analog shelf and that the Low (2) DCV terminal is connected via the MB channel to MEAS pin 2. This approach may prove easier to understand.

Table F-3
1796 TEST SYSTEMS
MULTIPLEXED DEVICE CHANNEL CONNECTIONS

Channel Name	Port No.	Terminal Name	Mnemonic	Device	Remarks
MA	1	T-1	HI	ACV	
	2	T-2	HI	FMU	Channel A (Pos. 6, J1) (1)
	3	T-3		MASCAN-3	EXT MEAS A (J5-Analog Shelf) (2)
	4	T-4		MASCAN-4	Spare Jack (Pos. 6, J2) (3)
	5	T-5	HI	DCV	DCV + (HI)
	6	T-6	GUARD	RMU	RMU Guard
	7	T-7	VP	RMU	RMU V+
MB	1	T-8		MBSCAN-1	Spare Jack (Pos. 8, J1) (1)
	2	T-9	LO	FMU	Channel B (Pos. 7, J1) (1)
	3	T-10		MBSCAN-3	EXT MEAS B (J6-Analog Shelf) (2)
	4	T-11		MBSCAN-4	Spare Jack (Pos. 7, J2) (1, 3)
	5	T-12	LO	DCV	DCV — (LO)
	6	T-13	GSENSE	RMU	RMU Guard Sense
	7	T-14	VM	RMU	RMU V —
SA	1	T-15		ACS-1	ACS-1 (or ACS) (Pos. 15, J2) (1)
	2	T-16		PGEN-2	PGEN-2 (Pos. 15, J1) (1)
	3	T-17		SASCAN-3	EXT SOURCE A (J3-Analog Shelf) (4)
	4	T-18	VP	RMU	RMU V+
	5	T-19	IP	RMU	RMU I+
SB	1	T-20		PGEN-1	PGEN-1 (or PGEN) (Pos. 17, J1) (1)
	2	T-21		ACS-2	ACS-2 (Pos. 17, J2) (1)
	3	T-22		SBSCAN-3	EXT SOURCE B (J4-Analog Shelf) (4)
	4	T-23	VM	RMU	RMU V —
	5	T-24	IM	RMU	RMU I —
(1) Rear of Analog Scanner Box. (2) XMEAS-1 also connects EXT MEAS A and EXT MEAS B. (3) XMEAS-2 also connects Pos. 6, J2 and Pos. 7, J2 spares. (4) XSOURCE-1 also connects EXT SOURCE A and EXT SOURCE B.					

Device specifications that contain both implicit (default) and explicit commands within the same specification are illegal and will result in a translation error. For example:

MEAS DCV AT(HI MA@1:2);
 Explicit Implicit

is an erroneous specification that will cause a translation error to occur.

Table F-3A

**1799 TEST SYSTEMS
MULTIPLEXED DEVICE CHANNEL CONNECTIONS**

<u>Channel Name</u>	<u>Port No.</u>	<u>Terminal Name</u>	<u>Mnemonic</u>	<u>Device</u>	<u>Remarks</u>
MA	1	T-1	HI	ACV	
	2	T-2	HI	FMU	Channel A ⁽¹⁾
	3	T-3		MASCAN-3	EXT MEAS A (POS 10, J2) ^(1,2)
	4	T-4		MASCAN-4	Spare Jack (POS 10, J1) ^(1, 3)
	5	T-5	HI	DCV	DCV + (HI)
	6	T-6	GUARD	RMU	RMU GUARD
	7	T-7	VP	RMU	RMU V+
MB	1	T-8		MBSCAN-1	Spare Jack (POS 11, J1) ⁽¹⁾
	2	T-9	LO	FMU	Channel B
	3	T-10		MBSCAN-3	EXT MEAS B (POS 11, J3) ^(1, 2)
	4	T-11		MBSCAN-4	Spare Jack (POS 11, J5) ^(1, 3)
	5	T-12	LO	DCV	DCV – (LO)
	6	T-13	GSENSE	RMU	RMU Guard Sense
	7	T-14	VM	RMU	RMU V–
SA	1	T-15		ACS-1	ACS-1 (ACS) (POS 7, J5) ⁽¹⁾
	2	T-16		SPARE	PGEN-2 (POS 7, J1) ⁽¹⁾
	3	T-17		SASCAN-3	EXT SOURCE A (POS 7, J3) ⁽⁴⁾
	4	T-18	VP	RMU	RMU V+
	5	T-19	IP	RMU	RMU I+
SB	1	T-20		PGEN-1	PGEN-1 (or PGEN)
	2	T-21		SPARE	ACS-2 (POS 8, J1) ⁽¹⁾
	3	T-22		SBSCAN-3	EXT SOURCE B (POS 8, J2) ^(1, 4)
	4	T-23	VM	RMU	RMU V–
	5	T-24	IM	RMU	RMU I–

(1) Rear of Analog Scanner Box.

(2) XMEAS-1 also connects EXT MEAS MA and EXT MEAS MB.

(3) XMEAS-2 also connects Pos 10, J1 and Pos 11, J5 spares.

(4) XSOURCE-1 also connects EXT SOURCE SA and EXT SOURCE SB

Pin specifications may also contain zeros (0) or asterisks (*). For example,

MEAS RMU GUARD AT (1:2:3:4);

MEAS RMU GUARD AT (1:2:0:0);

adding zeros to the last statement will **clear** the last two pins of that statement (scanner and MUX relays connected to the last two pins are opened). The asterisk (*), on the other hand, means "leave unchanged". For example,

MEAS RMU GUARD AT (1:2:8:9);

MEAS RMU GUARD AT (5:6:*:*);

would allow connections to remain **unchanged** (*) at the last two pins of the second statement while changing the first two pins.

Scanner statements that specify particular instruments (such as MEAS ACV, TEST DCI, etc.) make connections in the following manner:

1. The MEAS ACV, TEST DCV segment performs a measurement or test only. Connections to the MUX channels or scanner pins are **not** made. For example, the statements

MEAS DCV MAX=6;

TEST DCV < HI=2.7 LO=2.3 > [40] MAX=3;

are valid and would perform measurements or tests only. **No** connections to MUX channels or scanner pins are made. All pins remain in the state that they were in previously.

2. The AT construct connects or disconnects channels and/or scanner pins that are to change state.

3. Only **one** device can be connected to each channel at one time when using statements that specify specific devices (such as SET **DCV**, MEAS **ACV**, TEST **DCI**, etc.). For example, it would **not** be possible to connect PGEN-2 and ACS-1 to the SA channel at the same time. The statements

20: SET ACS-1 AT(1);

30: SET PGEN-2 AT(3) TD=2U TS=5U VD=5 V5=0;

would result in the attachment of ACS-1 to SOURCE Pin 1 on the analog shelf over the SA channel at label 20. Since PGEN-2 **also** uses the SA channel, ACS-1 will automatically be removed from that channel **before** PGEN-2 is connected at label 30.

4. For certain types of measurements, it may be advantageous to connect more than one device (instrument) to a channel at the same time. This can be done only with the mSCAN keywords and Terminal names (T-n) described in the following paragraph.

External Device Connections (mSCAN-n, XMEAS-n, XSOURCE-n)

Any of the devices, spare terminals, or external shelf jacks listed in Tables F-3 and F-3A, and Figures F-5 and F-5A may be accessed directly through use of the following four keywords.

MASCAN-n

MBSCAN-n

SASCAN-n

SBSCAN-n

For example, if the spare BNC jack for the MA channel were to be connected through to analog shelf pin 6, the following command could be issued:

SET MASCAN-4 AT(6);

Where -n represents one of the port numbers listed in Table F-3 and F-3A for the channel name specified (MA, MB, SA, SB). Any of the port numbers can be directly addressed through use of these mSCAN n keywords.

The following BNC connections are also available at the rear of the analog scanner box for connection of spare devices to any channel and direct access to the devices through use of the following mSCAN-n keywords.

1796 TEST SYSTEMS

Keyword	BNC Jack	Channel	Remarks
MASCAN-4	Pos. 6, J2	MA	Spare MA.
MBSCAN-1	Pos. 8, J1	MB	Spare MB.
MBSCAN-4	Pos. 7, J2	MB	Spare MB.
MASCAN-2	Pos. 6, J1	MA	Spare if FMU not used (Chan A).
MBSCAN-2	Pos. 7, J1	MB	Spare if FMU not used (Chan B).
SBSCAN-1	Pos. 17, J1	SB	Spare if PGEN-1 not used.
SASCAN-2	Pos. 15, J1	SA	Spare if PGEN-2 not used.
SASCAN-2	Pos. 15, J2	SA	Spare if ACS-1 not used.
SBSCAN-2	Pos. 17, J2	SB	Spare if ACS-2 not used.

1799 TEST SYSTEMS

Keyword	BNC Jack	Channel	Remarks
MASCAN-4	Pos. 10, J1	MA	Spare MA.
MBSCAN-1	Pos. 11, J1	MB	Spare MB.
MBSCAN-4	Pos. 11, J5	MB	Spare MB.
SASCAN-2	Pos. 7, J1	SA	Spare
SASCAN-1	Pos. 7, J5	SA	Spare
SBSCAN-2	Pos. 8, J1	SB	Spare

A SET mSCAN command issued **without** a -n port number assignment (for example, SET MASCAN) interconnects any pins specified in the AT construct via the channel named.

The following XSOURCE-1 and XMEAS-1 or -2 keywords are also available for connecting spare external devices that require **two** channels to the scanner system:

1796 TEST SYSTEMS

Keyword	Connects From/To:	Remarks
XMEAS-1	EXT A and EXT B MA MB	BNC Jacks on Analog Shelf. EXT A is HI and EXT B LO (MEASURE CHANNELS)
XMEAS-2	Spare Pos. 6, J2 and Spare Pos. 7, J2 MA MB	BNC Jacks at rear of analog scanner box. MA is HI and MB LO (SPARE MEAS JACKS)
XSOURCE-1	EXT A and EXT B SA SB	BNC Jacks on Analog Shelf. EXT A is HI and EXT B is LO (SOURCE CHANNELS)

1799 TEST SYSTEMS

Keyword	Connects From/To:	Remarks
XMEAS-1	EXT MA and EXT MB MA MB	BNC jacks at rear of Analog measurement Box. EXT MA (Pos. 10, J2) is HI EXT MB (Pos. 11, J3) is LO
XMEAS-2	Spare Pos. 10, J1 and Spare Pos. 11, J5 MA MB	BNC Jacks at rear of analog measurement box. MA is HI and MB LO. (SPARE MEAS JACKS)
XSOURCE-1	EXT SA and EXT SB SA SB	BNC jacks at rear of analog measurement box. EXT SA (Pos. 7, J3) is HI EXT SB (Pos. 8, J2) is LO (SOURCE CHANNELS)

If two MEAS channels are required for an external device connection for example, the following SET statement

```
SET XMEAS-1 AT(2:3);
```

could be used to connect the two analog shelf BNC jacks labelled EXT A and EXT B through the MA (HI) and MB (LO) channels to analog shelf MEAS pins 2 (MA) and 3 (MB).

The following notes are applicable to statements that include the keywords mSCAN-n, XMEAS-1 or -2 and XSOURCE-1:

1. The port numbers (-n)- listed in Table F-3 and F-3A are used with the MASCAN, MBSCAN, SASCAN, and SBSCAN keywords only. The XMEAS-1, XMEAS-2, and XSOURCE-1 keywords must include the -1 or -2 numbers shown. Only the numbers shown can be used with XMEAS or XSOURCE keywords.

2. A channel can be cleared (MUX and scanner relays are opened) with any of the following commands:

Command	Function
CLEAR MASCAN	Clears MA channel only.
CLEAR MBSCAN	Clears MB channel only.
CLEAR SASCAN	Clears SA channel only.
CLEAR SBSCAN	Clears SB channel only.
CLEAR SCAN	Clears all SA, SB, MA, MB and power supply channels.
CLEAR XMEAS-1	Clears MA and MB channels only.
CLEAR XMEAS-2	Clears MA and MB channels only.
CLEAR XSOURCE-1	Clears SA and SB channels only.
CLEAR ALL	Clears all devices and connections.

3. Multiple devices (instruments) can be connected to the **same** MUX channel at the same time through use of the MASCAN, MBSCAN, SASCAN, and SBSCAN keywords **only**. For example, the statements

```
30: SET FMU TRIGA=2.4+ TRIGB=1.2;
40: SET MASCAN AT (T-5, T-2 @ 1);
50: SET MBSCAN AT (T-12, T-9 @ 2);
60: MEAS FREQ;
70: MEAS DCV MAX=5;
```

would connect the HI side of the DCV (T-5) and the HI side of the FMU (T-2) together on the MA channel at label 40. The LO side of the DCV (T-12) and the LO side of the FMU would then be connected together in the next statement and would remain connected for the measurements that follow. Any or all of the T n numbers listed in Table F-3 and Table F-3A for the individual channel designated by the mSCAN keyword may be listed in AT constructs to connect more than one device to the same channel.

4. Connecting more than one channel to the same scanner pin (Figure F-6) is permitted. Care **must** be exercised, however, as current sources can damage the dry relay scanner contacts if they are switched with power applied. Current sources may also damage the driver/sensors, if connected.

CAUTION

Connecting or disconnecting channels to circuits while current is applied can damage scanner relay contacts or driver/sensors (if connected).

For example, listing a pin in an ASSIGN LGC list **will** close the associated hybrid scanner D/S relay for that pin. Since all four channels (MA, MB, SA, SB) may **also** be connected to the **same** pin at the same time (if the pin is also listed in scanner device specification pin lists) care **must** be exercised to avoid connecting devices that may be current sources (such as the ACS, PGEN, etc.). These devices could destroy the driver/sensors or the scanner relay contacts, if the relays are switched with power applied.

Hybrid/Analog Scanner Pin Assignments

Multiplexed channels (MA, MB, SA, SB) may be assigned (scanned) to separate SOURCE (for SA, SB) or separate MEAS (for MA, MB) pins on the analog shelf and/or to D/S pins on the digital shelf through the analog and/or hybrid pin scanner system (see Figure F-6).

Up to 36 separate analog MEAS and 36 separate analog SOURCE shelf pins can be connected through the analog pin scanner to the multiplexed MA, MB, SA, SB channels. Each channel is connected through groups of 12 scanner relays which interconnect up to 12 shelf pins per group.

Up to 192 D/S pins may also be assigned through the hybrid pin scanner to the MA, MB, SA, SB channels. Each channel is connected through groups of 48 scanner relays which interconnect up to 48 shelf pins per group.

Hybrid and analog scanner channel groups that are not required are disconnected at the MUX in order to minimize cross-talk. Driver/Sensors are connected through the hybrid scanner to the D/S shelf pins whenever the shelf pins are included in ASSIGN LGC pin lists. The multiplexed channels (MA, MB, SA, SB) are connected through the hybrid and/or analog scanner to D/S shelf pins and/or analog shelf pins whenever the following set of tokens are used to specify scanner pin lists:

Token	Function
HYBRID PIN SCANNER (1796/1799)	
H=	Analog pin only. This pin should not have been previously assigned as a D/S pin.
C=	Analog and/or Digital Pin. This pin can be assigned as an analog pin concurrently with its usage as a D/S pin.
ANALOG PIN SCANNER (1796 only)	
A=	Assign to Analog pin scanner only (not Hybrid). Pins assigned by the A= token can appear anywhere in the list.
Blank*	Pins listed without tokens are automatically assigned to the Analog pin scanner. Pins without tokens must appear first in the list.

For example, the statement

```
SET MASCAN-4 AT(1, 2, H=3, 4);
```

would assign the spare BNC jack at the rear of the analog box (Pos. 6, J2) through the analog scanner to analog shelf MEAS pins 1 and 2. It would also assign the same jack through the hybrid scanner to digital shelf D/S pins 3 and 4.

The statements (1) through (3)

- ```
(1) SET MASCAN-4 AT(C=5, 6, A=7, 8);
 or
(2) SET MASCAN-4 AT(7, 8, C=5, 6);
 or
(3) SET MASCAN-4 AT(7, 8, H=5, 6);
```

perform identical functions, and will all assign Jack J2 to pins 7 and 8 on the analog shelf, and to pins 5 and 6 on the digital shelf. The following use differences should be noted, however:

1. The A= token can be placed **anywhere** in list (1) while pins **without** tokens **must** appear **first** in the list as indicated at (2) and (3).
2. The C= token at (2) is identical to the H= token at (3) with the exception that pins listed by the C= token may **also** be assigned as driver/sensor pins if the same pin numbers are included in ASSIGN LGC statements. If pins listed by the H= token are **also** included in ASSIGN LGC lists, a translation error will occur as they have been assigned as analog pins **only**.

If the C= token and the H= token are **both** included in the **same** pin list, then **all** pins listed by both tokens **will** be assigned as C= pins. For example, the statement

```
SET MASCAN-4 AT(C=3, 4, H=5, 6);
```

\*In 1799 systems blank pin assignments are not permitted. An error message will result when the DIAGNOSIS Mode is entered for UUT testing.

would assign **all** pins (3, 4, 5, 6) as C= pins that could also be connected to driver/sensors.

### Relay Protection (RELPRO)

In order to provide steady-state connections, digital and analog shelf pins, controlled by the hybrid and analog scanner relays, normally remain connected **if** pins in the old and new incremental statements do not change. For example, in the following two AT device specifications:

```
AT(1-4, 20, 27);
```

```
AT(4, 8, 27);
```

Analog shelf pins 4 and 27 do not change state. The normal order of MUX and scanner relay operation would be as follows when the second statement is processed:

1. The MUX relay for analog pin group 13-24 opens because relays in this group will **not** be used in the second pin specification.
2. The MUX relays for analog pin groups 1-12 and 25-36 **cannot** open because pins 4 and 27 are identical in both specifications and therefore must remain in a steady state. Note that if the identical pins (4 and 27) in the second statement had been assigned instead to the **same** group of 12 pins (for example, pins 3 and 4) then the MUX relay controlling pins 25-36 could also have been opened at this time because pins in this group would not have been used in the second specification.
3. Next, the analog scanner relays controlling analog shelf pins 1, 2, 3, and 20 are opened as these pins are not used in the second specification.
4. Next, the analog scanner relay controlling analog shelf pin 8 is closed as this is a new pin that is to be used in the second specification.

As a general rule, the procedure is as follows:

1. Open all MUX group relays that are **not** to be used again.
2. Open all hybrid or analog scanner pin relays that are not to be used again.
3. Close all hybrid or analog scanner pin relays that are new.
4. Close all MUX group relays controlling pins that are to be used and are not already closed.

Note that in the example just described, analog scanner relays controlling pins 1, 2, and 3 were opened and the relay for pin 8 was closed while the group relay for pins 1-12 was still connected to the MUX, because pin 4 was identical and therefore had to be kept in a steady state. This was also true of the group relay for pins 25-36, however, shelf pins in this group did not change state. If a high current source had been across any of the scanner pin relays in group 1-12 as they were switched, damage to the relay contacts **could** have occurred.

The keyword RELPRO (Relay Protect) can be inserted into a device specification and will cause all MUX group relays associated with scanner pins that change state to open prior to switching the scanner pins in or out as added relay protection. For example, the specification

```
AT (RELPRO @ 4, 8, 27);
```

would cause relays to operate in the following sequence:

1. The MUX relay for analog pin group 1-12 opens because pins 1, 2, and 3 within the group change state and the RELPRO keyword has been added.

2. The MUX relay for group 13-24 opens because pins in this group are not to be used again. Note that the MUX relay for group 25-36 does **not** open (even when using RELPRO) because the **only** pin listed in this group (pin 27) must remain in a steady state and no other pins in the group change state. If a pin **had** changed state within the group, then the MUX relay for group 25-36 would also have been opened.

3. Next, the scanner relays for pins 1, 2, 3, and 20 are opened, and the relay for pin 8 is closed.

4. Next, the MUX group relay for group 1-12 is closed as pins within this group are to be used again.

In general, the keyword RELPRO protects the scanner relay contacts from source currents by opening any group heavy duty MUX relays associated with pins that **change** state prior to switching the scanner relays. After switching the scanner relays, the MUX relays are closed again.

### Unscanned Shelf Connections (SHELF)

The ACS and PGEN source outputs may be distributed (under program control) to dedicated (unscanned) pins at connectors J24 and J28 on the 1796 analog shelf, and J30 on the 1799 Driver/Sensor Unit (see Figures 5 and 5A).

The keyword SHELF **must** be added to a device specification in order to connect the unscanned pins. For example, the statement

```
SET ACS-1 AT (SHELF@1);
```

will assign ACS-1 to analog shelf SOURCE pin 1 and at the same time to unscanned pins at J24 on the 1796 analog shelf while the statement

```
SET ACS-1 AT (SHELF@C=1);
```

will assign ACS-1 to hybrid pin 1 and at the same time to unscanned pins J24 (1796) or J30 (1799).

The statement: 

```
SET ACS-1 AT (SHELF);
```

will now disconnect the ACS-1 from scanned analog SOURCE pin 1 but leave it connected to the unscanned pins on J24 (1796) or J30 (1799).

```
SET ACS-1 AT(1); or SET ACS-1 AT(C=1);
```

will now disconnect the ACS-1 from the unscanned pins at J24 (1796) or J30 (1799) and re-connect it to the scanned SOURCE pin 1 on the analog shelf, or hybrid pin 1.

## F2.3 SCANNER INSTRUMENT PROGRAMMING

### F2.3.1 Frequency Measurement Unit (FMU)

The FMU has two input measure channels (A and B) that may be connected to scanner pins through MUX channels MA and MB respectively by use of scanner language statements. The FMU is programmed using the procedures described in Para. 3.2.14. While the statements shown in Para. 3.2.14 may be used, it is also permissible to include any of the explicit 1796 scanner language constructs for greater clarity. For example, the following statement

```
SET FMU AT(HI MA @ 3:LO MB @ 24) TRIGA=2.4 + TRIGB=1.2;
```

is identical to the one described in Para. 3.2.14 but is explicit and thus makes it easier to understand that the FMU Channel A (HI) is connected through the MA channel to analog shelf MEAS pin 3 and that FMU Channel B (LO) is connected through the MB channel to MEAS pin 24.

### F2.3.2 Pulse Generators (PGEN-1, PGEN-2)

In 1796 test systems up to two pulse generators may be programmed through the SB channel for (PGEN-1) and SA channel (for PGEN-2) to hybrid and/or analog shelf scanner pins. A single Pulse Generator (PGEN-1) is supplied with the 1799 test systems.

Methods of programming the pulse generators are described in para. 3.2.16 with the following exceptions:

1. Any of the explicitly 1796/1799 scanner language constructs may be added for clarity, if desired.
2. The keyword SHELF, described in para. F2.2.3 **must** be used to connect the **unscanned** analog shelf pins at J24 (1796), J30 (1799) test systems. For example,

SET PGEN-1 AT(SHELF@5)TD=2U TS=5U VD=5 V5=0;

or

SET PGEN-1 AT(SHELF@C=5)TD=2U T5=5U VD=5 V5=0;

will also connect the unscanned pins at J24 in 1796 test systems or J30 in 1799 test systems in addition to scanned analog/hybrid SOURCE pin 5.

#### NOTE

In standard 1799 systems a single pulse generator (PGEN-1) is provided. A second pulse generator (PGEN-2) is supported by the software.

### F2.3.3 AC Sources (ACS-1, ACS-2)

In 1796 test systems up to two AC sources may be programmed through the SA channel (for ACS-1) and SB channel (for ACS-2) to hybrid and analog shelf scanner pins. A single AC source (ACS-1) is supplied with 1799 test systems. Methods of programming the AC Sources are described in para. 3.2.13 with the following exceptions:

1. Both Wavetek 154 and Wavetek 157 programmable waveform generators may be included as AC Sources. The manufacturer's instruction manuals should always be consulted prior to using either instrument.
2. Any of the explicit 1796/1799 scanner language constructs may be added for clarity, if desired.
3. The keyword SHELF, described in para. F2.2.3 **must** be used to connect the unscanned analog shelf pins at J28 (for ACS-1) or J24 (for ACS-2) in 1796 test systems or J30 (for ACS-1) in 1799 test systems. For example,

SET ACS-1 AT(SHELF@13);

or

SET ACS-1 AT(SHELF@C=13);

would also connect the unscanned pins at J28 in 1796 test systems or J30 in 1799 test systems in addition to scanned analog SOURCE pin13.

#### NOTE

In standard 1799 test systems a single AC Source (ACS-1) is provided. A second AC Source (ACS-2) is supported by the software.

### F2.3.4 Precision Multimeter (ACV, DCV/DCI, RMU, DCVIF)

The precision multimeter consists of a basic unit that can measure both dc volts (DCV) and dc current (DCI). Optional additions allow ac voltage (ACV), and resistance (RMU) to be measured. In addition, a high compliance current source option (DCVIF) to the RMU can be used for non-linear network measurements when desired (see Figure F-7). The unit is fully programmable over the ranges listed in Table F-4 and offers the following novel features:

1. High accuracy direct dc measurements with 16-bit resolution.
2. Four programmable measurement modes (direct, LPF, INTG, and nPEAK).
3. A track and hold feature that allows an external start signal (EXTS) to initiate a measurement.
4. Floating RMU measurements.
5. An RMU GUARD feature that reduces the effect of measurement errors resulting from resistive network influences.
6. Automatic ranging from a starting point as well as from the least sensitive range when MAX= is specified.

The various instruments that make-up the precision multimeter are connected to the multiplexed scanner MA, MB, SA, SB channels using the scanner language constructs described in Para. F2.2.3. In addition, the multi-meter keywords listed in Table F-5 may also be invoked by selected measurement modes (DCV, DCI, RMU, etc.) to perform the tasks indicated in the table. Some keywords (such as LOWZ) are used by only one or two of the measurement modes and are described further in the description of that mode. Other keywords (such as EXTS, FORCE, etc.) are used by a majority of the measurement modes to perform the following tasks

Table F-4

| 1796 MULTIMETER RANGE TABLE |                 |             |
|-----------------------------|-----------------|-------------|
| ACV                         | DCV             | DCI         |
| Max Range                   | Max Range       | Max Range   |
| 150V rms                    | 163.84V         | 163.84 mA   |
| 15V rms                     | 40.96V          | 40.96 mA    |
| 1.5V rms                    | 10.24V          | 10.24 mA    |
| 0.15V rms                   | 2.56V           | 2.56 mA     |
| 0.015V rms                  | 0.640V          | 640 $\mu$ A |
|                             | 0.160V          | 160 $\mu$ A |
|                             |                 | 40 $\mu$ A  |
|                             |                 | 10 $\mu$ A  |
| RMU (1, 4)                  |                 | DCVIF (3)   |
| Max Range (Ohms)            | Max I/V Force   | I Force     |
| 40                          | 64 mA           | 2 mA        |
| 160                         | 16 mA           | 1 mA        |
| 640                         | 4 mA            | 100 $\mu$ A |
| 2.56K                       | 1 mA            | 10 $\mu$ A  |
| 10.24K                      | 250 $\mu$ A     |             |
| 40.96K                      | 62.5 $\mu$ A    |             |
| 163.8K                      | 15.625 $\mu$ A  |             |
| 655.4K                      | 3.90625 $\mu$ A |             |
| 13,110 MEG                  | 4V (2)          |             |

- (1) When making a RMU measurement of less than 655.4K $\Omega$  the 2.56 V DCV range is automatically invoked.  
 (2) If a resistance value above 655.4K is being measured by the RMU, the 4V V Force mode is entered.  
 (3) DCVIF forces a current while measuring voltage at the input to the multimeter.  
 (4) Resistance measurements above 200K should also use LPF or INTG to reduce noise pick-up.

### DLY=n

This is a user specified delay of up to 1023 seconds, maximum, that is executed between the time that the instrument is connected to the UUT and measurement initiation. It has a resolution of  $\pm 5 \mu$ s. If a DLY is **not** listed (or n is set to zero) a default value of 10 ms is inserted. Delays are used by all measurement modes (DCV, DCI, ACV, RMU, DCVIF). The following statement

MEAS DCV AT(2:7)DLY=20M;

will invoke a delay of 20 ms between connection to the UUT and measurement initiation. The programmed delay can allow settling time when it is known that the UUT circuit requires it. Large number values entered for n with any multi-meter keyword may be expressed in engineering units as follows:

|     |            |
|-----|------------|
| MEG | for Mega-  |
| K   | for Kilo-  |
| M   | for Milli- |
| U   | for Micro- |

Table F-5

## MULTIMETER KEYWORDS

| Keyword              | Function                                                                                                                                                                                                                                                                                                                                                         | Used By:                       |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| DLY=n                | User specified delay (up to 1023 sec., max.). A delay is executed between the time that the instrument is connected to the UUT and measurement initiation. DLY defaults to 10ms, if omitted.                                                                                                                                                                     | DCV/DCI<br>ACV<br>RMU<br>DCVIF |
| EXTS                 | VOLTMETER START. The instrument will be armed to perform a designated measurement, however, the measurement will <b>not</b> start until a VOLTMETER START signal occurs (for track and hold operations).                                                                                                                                                         | DCV/DCI<br>RMU<br>DCVIF        |
| FORCE                | Suppress automatic range selection and <b>force</b> the MAX= range specified (or a default range) to be used.                                                                                                                                                                                                                                                    | ACV<br>DCV/DCI<br>RMU<br>DCVIF |
| INTG                 | Invokes the power line frequency integrate mode to remove noise due to power line pick-up.                                                                                                                                                                                                                                                                       | DCV/DCI<br>RMU<br>DCVIF        |
| LPF                  | Invokes the Low Pass Filter Mode to remove ac components not power line related or HF noise. The default range change delay of 10ms is increased to 210 ms when LPF is invoked.                                                                                                                                                                                  | ACV<br>DCV/DCI<br>RMU<br>DCVIF |
| LOWZ                 | Causes a Low Impedance (50 Ohm) Input Termination to be switched into the ACV. This feature is recommended for measurements at signal frequencies greater than 1 MHz to improve impedance matching in the cabling system. Note that the 15V and 150V ranges automatically disconnect this termination to protect it against burn-out.                            | ACV                            |
| MAX=n                | Allows user to set range of instrument by specifying the expected maximum measurement value (n) in terms of voltage, resistance, current, etc. Automatic ranging occurs if measurement can be more accurately made on another range and the keyword FORCE has not been used. If a maximum range is not specified, the highest range available is used initially. | DCV/DCI<br>RMU<br>DCVIF<br>ACV |
| NEGATIVE             | Reverses polarity of RMU current or voltage force during resistance measurements so that a second measurement can be made to cancel out any thermal emf's caused by relay closures.                                                                                                                                                                              | RMU                            |
| NPEAK<br>or<br>PPEAK | Invokes the peak detection mode to detect maximum positive (PPEAK) or negative (NPEAK) peaks of waveforms.                                                                                                                                                                                                                                                       | DCV/DCI                        |
| PRESET               | Use the keywords set-up by the <b>last</b> SET statement in <b>this</b> MEAS or TEST statement.                                                                                                                                                                                                                                                                  | DCV/DCI<br>RMU<br>DCVIF        |

Table F-5 (Cont'd)

| Keyword | Function                                                                                                                                                                                                                                                                                                                                                                              | Used By:       |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| RNG=n   | Included for compatibility with 1792D test systems. Converts 1792D X10 range table <b>code numbers</b> for DCV/DCI to the closest X4 1796 range available and selects this range for use. The ACV code numbers are used directly (no conversion is necessary). Cannot be used with the RMU or DCVIF. The =n value represents 1792D code numbers only ( <b>not</b> 1796 code numbers). | ACV<br>DCV/DCI |
| SLOW    | Lengthens the ACV detector time constant so that measurements at frequencies less than 150 Hz do not cause variations in the reading due to signal ripple.                                                                                                                                                                                                                            | ACV            |
| ZERO    | Cancels voltages offset within the DCV module from the actual measurement thus enabling more accurate results to be obtained. When ZERO is used, the DCV execution time is slowed somewhat because of the additional internal measurement required.                                                                                                                                   | DCV            |
| I=n     | A floating point value (or register) that contains a forcing current value (n). This value selects the DCVIF forcing current to be used and <b>must</b> be within $\pm 5\%$ of <b>one</b> of the following values or a software error will occur:<br><br>2 mA<br>1 mA<br>100 $\mu$ A<br>10 $\mu$ A                                                                                    | DCVIF          |

## EXTS

This keyword may only be inserted in SET statements (not MEAS or TEST) in order to set-up the selected mode (DCV, DCI, RMU, DCVIF) to perform track and hold operations. When a MEAS or TEST statement does occur, the instrument will be armed, but will not make the actual measurement **until** an external event triggers it. The external event must be a TTL low level applied to the VOLTMETER START BNC jack (J2) on the 1796 analog shelf or connector J30 on the 1799 Driver/Sensor unit. For example, the statements

```
SET DCV AT(2:7)EXTS MAX=5;
```

```
MEAS DCV PRESET ZERO;
```

allow the DCV to be set-up for a measurement that will not actually be made **until** the external triggering event (EXTS) occurs. Note that it was **not** necessary to include an AT construct in the MEAS statement as connections were established by the SET statement and did not change. Operating time can always be saved by **not** including AT constructs unless they are actually needed to change scanner and/or MUX connections.

## FORCE

The keyword FORCE is used by **all** measurement modes to suppress automatic range selection and **force** the MAX=n range specified (or a default range) to be used. When FORCE is included in SET, MEAS or TEST statements, the measurement **must** be made on the specified or default range. For example, the statement

```
MEAS DCV AT(2:7)MAX=35 FORCE;
```

will **force** the measurement to be made on the 40.96V DCV range even if an overload occurs (refer to Table F-4). In a similar manner, the statement

```
MEAS ACV AT(3)MAX=14 FORCE;
```

will **force** the measurement to be made on the 15V ACV range, even if an overload occurs. FORCE may be useful under any of the following conditions:

1. If a signal that varies with time is being measured and displayed, it is often more convenient to have a test loop executed in a consistent fashion by forcing a known range.
2. For certain measurements that are used for computing gains, ratios, etc., any systematic errors of the measurement modules are often reduced by using the same range.
3. Knowing the expected result (the usual case for a test) causes the system to operate faster.

The following notes are applicable when the keyword FORCE is used:

1. If the forced range is inappropriate for the particular measurement a software error (overload) will occur if the measured signal is **greater** than the forced range. If the best range is **less** than the forced range, a less accurate measurement will be made.
2. The use of FORCE generally results in a faster execution time, and is therefore useful in cases where the user is relatively sure of the measurement range to be forced.
3. The keywords EXTS, NPEAK, and PPEAK (used with DCV, DCI measurements) automatically invoke FORCE.
4. If MAX=n is **not** used to specify the initial range to be used, then the highest range available will be the first trial range if the keyword FORCE is included. For compatibility with 1792D test systems, any range established by RNG=n will also be the forced range.

## INTG

The keyword INTG is used in all measurement modes except ACV to invoke the power line frequency integrate feature. This feature allows the rejection of superimposed noise (for example 50 or 60 Hz hum) at power line frequencies. It provides a noise rejection of greater than 40 dB over one power line period (16.67 ms for 60 Hz or 20 ms for 50 Hz). For example, the statement

```
MEAS DCI AT(2:7)INTG MAX=10M;
```

will use the DCI to make an integrated current measurement on the 10.24 mA range.

## LPF

This keyword is used in all measurement modes to invoke the low pass filter. The filter rejects (filters-out) high frequency noise superimposed on signals that are not power line related, and provides a noise rejection of greater than

20 dB at power-line frequencies. For example, the statement

```
MEAS DCV AT(2:7)MAX=8 LPF;
```

uses the DCV to make a filtered measurement on the 10.24 V DCV range. Note that if LPF is included in an ACV statement, that the filtering occurs after the ac-dc conversion, just prior to the dc-to-digital conversion.

## MAX=n

This keyword is used in all measurement modes to establish an initial measuring range. The value n is set to the maximum expected value of the measurement to be made. It allows a user to set the instrument range without reference to code numbers or range parameters. Only a knowledge of the maximum expected UUT value to be measured is required. For example, the statement

```
MEAS DCV AT(2:7)MAX=12;
```

will cause the 40.96V DCV range to be selected (refer to Table F-4). The software automatically selects the correct range for the value (listed in terms of volts, ohms, etc.) specified by MAX=n. The multimeter is switched to another range if the measurement can be more accurately made on a different range (or if an overrange has occurred) and the keyword FORCE has not been used to force a specific range.

If a MAX=n range is not specified in a SET, MEAS, or TEST statement, then the **highest** range available for that measurement mode is used initially. For example, the statement

```
MEAS DCV AT(2:7);
```

is permissible, and will cause the DCV to start measuring on the highest range available (163.84V for the DCV). If the value to be measured is at the low end of the scale (for example 2V), then the meter will automatically downrange until the correct range is established. Note that downranging can jump more than one range at a time when appropriate. Time can usually be saved by specifying the maximum value to be measured using MAX=n. Values for n may be expressed in engineering units as follows:

|     |            |
|-----|------------|
| MEG | for Mega-  |
| K   | for Kilo-  |
| M   | for Milli- |
| U   | for Micro- |

## PRESET

This keyword may be inserted in MEAS or TEST statements only (**not** SET statements). PRESET allows the **last** SET statement (for all measurement modes except ACV) to invoke the following keywords:

| DCV/DCI | RMU      | DCVIF |
|---------|----------|-------|
| DLY=    | DLY=     | DLY=  |
| EXTS    | EXTS     | EXTS  |
| FORCE   | FORCE    | FORCE |
| INTG    | INTG     | INTG  |
| LPF     | LPF      | LPF   |
| MAX=    | MAX=     | MAX=  |
| NPEAK   | -        | -     |
| PPEAK   | -        | -     |
| RNG=    | -        | -     |
| -       | NEGATIVE | -     |
| -       | -        | I=    |

It is therefore possible to set-up any of the keywords listed above in a SET statement that is inserted ahead of the MEAS or TEST statement containing the keyword PRESET. For example, the statements

```
SET DCV AT(2:7)MAX=12 LPF EXTS FORCE;
.
.
MEAS DCV PRESET;
```

will set-up the MAX=12, LPF, EXTS, and FORCE keywords which will then be invoked by the keyword PRESET in the MEAS statement that follows.

If PRESET is inserted into a MEAS or TEST statement that is **not** preceded by another SET statement a software error will occur. The keywords **must** be set-up in a SET statement prior to invoking PRESET in a MEAS or TEST statement.

PRESET is ideal for use in adjustment loops where connections and parameters can be established once by the SET statement and then used again by the MEAS or TEST statement that follows. For example, the statements

```
40: SET DCV AT(4:5)MAX=5;
50: WRITE DEV=TEC'
 TURN R-28 FULLY CCW,
 THEN PRESS CONTINUE';
60: PAUSE;
70: WRITE DEV=TEC'
 TURN R-28 CW ¼ TURN,
 THEN PRESS CONTINUE';
80: PAUSE;
90: TEST DCV < HI=5 LO=3 > [70] PRESET;
.
.
```

establish an adjustment loop for R-28. The parameters and connections are established by the SET statement at label 40, rather than the TEST statement at label 90 thus increasing operating efficiency, as these parameters do not have to be set-up each time that the loop is executed, if PRESET is used.

It is also possible to have multiple MEAS or TEST statements with PRESET that refer back to the same SET statement that sets-up the keywords. For example, the following statements

```
(1) SET DCV AT(2:7)MAX=12 LPF;
.
.
(2) MEAS DCV PRESET;
.
.
(3) MEAS DCV PRESET;
```

establish connections and set-up the keywords MAX=n and LPF at (1) then invoke them at (2) and (3).

The following paragraphs describe each of the various measurement modes available (ACV, DCV, DCI, RMU, DCVIF), and further define some of the more selective keywords.

## AC Voltage Measurements (ACV)

The ACV Mode of the precision multimeter is entered whenever ACV is specified in MEAS, or TEST statements (SET statements are not used). Methods of programming the ACV are described in Para. 3.2.12 with the following exceptions:

1. The following multimeter keywords may be used with ACV MEAS or TEST statements (refer to Table F-5): Where branches occur, any one of the paths shown may be taken as the MEAS or TEST path is followed from left (start) to right (end).



2. When using the ACV, either MAX=n or RNG=n may be specified, or both can be omitted and the range will default to the highest available range (150V rms). MAX=n specifies the expected maximum measurement value in terms of voltage and is the recommended method of range setting as it does not require a user to memorize the relationship between range code numbers and actual voltage ranges. For example, the statements

MEAS ACV AT(8)MAX=100;

or

MEAS ACV AT(8)RNG=1;

are identical and will both establish the 150V rms range (1792D Code No. 1) however, the first statement allows a user to specify the maximum voltage to be measured and the software automatically sets itself to that range. Automatic ranging will then occur if the measurement can be more accurately made on another range. The RNG=n keyword and 1792D code numbers may be used, however, to set the ACV directly to a specified ACV range without conversion, if desired.

3. The keywords SLOW and LOWZ are described in Para. 3.2.12. SLOW sets the ACV time constant so that measurements at frequencies below 150 Hz can be made, while LOWZ invokes a low-impedance (50 Ohm) input termination for measurements. The use of LOWZ is recommended above 150 kHz. Note that the 15 V and 150 V ranges automatically disconnect this termination to protect it against burnout.

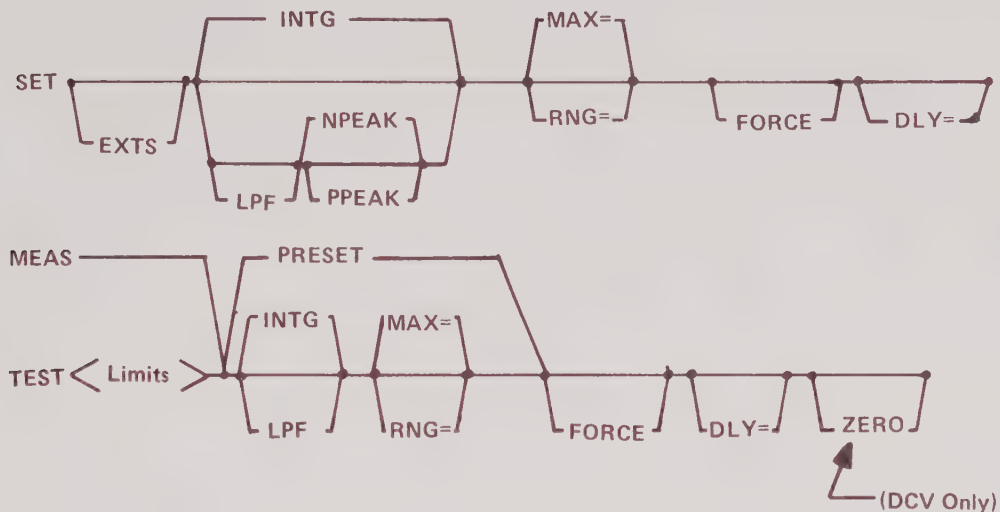
## DC Voltage/current Measurements (DCV/DCI)

The DCV and DCI modes of the precision multimeter are entered whenever DCV or DCI are specified in SET, MEAS, or TEST statements. Scanned shelf pins may be connected to two input terminals (HI and LO) on the MA or MB channels using any of the scanner language constructs described in Para. F2.2.2 (see Figure F-7). For example, the statement

SET DCV AT(2:7);

will automatically connect the HI (+) DCV (or DCI) input through the MA channel to analog shelf pin 2, and the LO (—) input through the MB channel to analog shelf pin 7.

The following multimeter keywords listed in Table F-5 may be used with DCV/DCI SET, MEAS, or TEST statements:



Follow each SET, MEAS, or TEST path from left (start) to right (end) through the diagram.

Where branches occur, any **one** of the paths may be taken. Note, however, that it is **not** possible to include the keywords LPF, NPEAK, or PPEAK in a SET statement if the keyword INTG is used. Note also that EXTS is used with SET statements only, and that the keyword ZERO applies to the DCV only (not the DCI). Any of the keywords in a SET statement can be invoked by the keyword PRESET in a MEAS or TEST statement that follows. Note, however, that it is not possible to include INTG, LPF, MAX= or RNG= in a MEAS or TEST statement when PRESET is invoked.

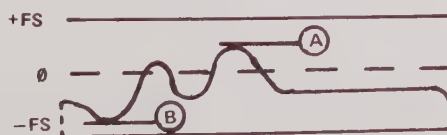
The keywords NPEAK and PPEAK set-up the peak detection feature when inserted in SET statements. This feature is ideal for detecting pulse or ac waveform peaks when using either the DCV or DCI measurement modes. SET statements set-up (arm) the detector which is then automatically triggered when the **next** MEAS or TEST statement using the keyword PRESET occurs. For example, the statements

(1) SET DCV AT(2:7)MAX=6 LPF PPEAK;

(2) MEAS DCV PRESET DLY=20M;

will set-up (arm) the peak detector at (1) so that the next MEAS or TEST statement (2) can trigger the measurement. NPEAK, PPEAK, and EXTS automatically invoke FORCE.

Both negative (NPEAK) and positive (PPEAK) signal peaks may be detected by the peak detector, where PEAK is the most positive (or most negative) algebraically going signal peak detected during the measurement. Peaks are measured algebraically as the following example illustrates:



A is the most positive going (PPEAK) detected while B is the most negative going (NPEAK) peak detected.

Note that it is possible to obtain a negative value for a positive (PPEAK) measurement or a positive value for an NPEAK measurement if the peak does **not** cross the zero threshold. It is also possible to obtain a zero result if no signal input is detected.

Low duty cycle pulse peaks are captured by charging-up a capacitor, then removing the charging source. A minimum charging signal duration of 10  $\mu$ s is required for a 90% reading while 30  $\mu$ s provides 98% detection ( $\pm 2\%$ ) for signals ranging from dc to 40 KHz. The capacitor is automatically discharged whenever a SET statement is encountered.

The DLY=n keyword may be used with NPEAK and PPEAK to increase the delay between connecting the DCV/DCI for measurement and measurement initiation (which for NPEAK and PPEAK is accomplished by inserting the keyword EXTS in a SET statement and then triggering the meter with an external VOLTMETER START signal). If DLY=n is omitted, a default delay of 10  $\mu$ s is assumed. Measurement of unwanted signals may therefore be eliminated by increasing the delay set-up time so that the unwanted part of the signal passes before a measurement is actually made on the desired part.

Signals requiring a long time span may be measured by inserting a standard DELAY statement into the test program. For example, the statements

```
30: SET DCV AT(1:2)MAX=6 PPEAK EXTS DLY=20M;
40: DELAY 10M;
50: MEAS DCV PRESET;
```

set-up the DCV for a PPEAK measurement at label 30. Next, they provide a 10 ms delay (after the VOLTMETER START external signal occurs) at label 40.

The keyword ZERO is used with DCV only (not DCI) to cancel any voltage offset from the actual measurement thus enabling very accurate results to be obtained. For example, the statement

```
MEAS DCV AT(2:7)MAX=3 ZERO;
```

would invoke the ZERO feature to cancel any voltage offset from the measurement. When ZERO is used, the DCV execution time is slowed somewhat because of the additional measurement time required. Note that the zeroing function is for the module only. Offsets in the cabling system are not accounted for.

TEST statements allow measured DCV/DCI results to be stored in the DCV (or DCI) register and then compared against the limits (limits) given in the statement. For example, the statement

```
290: TEST DCV AT(2:7) (HI=2.7 LO=2.3) [330] MAX=3;
```

will make the comparison, and then branch to label 330 if the test conditions shown (HI=2.7 LO=2.3) are not met. Conditional error branches may be used with both MEAS and TEST statements. A MEAS branch will occur if the measurement cannot be made.

The following commands can be used to clear DCV/DCI connections:

| Command   | Function                                   |
|-----------|--------------------------------------------|
| CLEAR DCV | Clears (disconnects) DCV.                  |
| CLEAR DCI | Clears (disconnects) DCI.                  |
| CLEAR ALL | Clears <b>all</b> devices and connections. |

### Resistance Measurements (RMU)

The RMU mode of the precision multimeter is entered whenever RMU is specified in SET, MEAS, or TEST statements. The RMU 1796 configuration is capable of performing two terminal guarded, two terminal unguarded, and four

terminal measurements over a wide resistance range. Figure F-9 shows the various measurement connection configurations available. The four terminal unguarded measurement provides the most accurate results because voltage sense and current leads can be Kelvin-connected together at the shelf device adaptor (refer to Section 2 of the Operating Instructions). Two terminal guarded and two terminal-unguarded (non-Kelvin) measurements are made with the voltage sense and current leads connected together at the MUX, and are used whenever the accuracy of a four terminal measurement is not required or when guarded measurements are to be made.

Table F-4 lists the range of resistances that can be measured. Resistance values below 655.4K $\Omega$  are measured by forcing a known current through the unknown resistance ( $R_x$ ) and then measuring the voltage. Resistances above 655.4K $\Omega$  are measured by forcing a known voltage (4-V) across the unknown resistance and then measuring the current.

Connections may be made to scanned pins on the digital or analog shelves or directly to unscanned pins at the analog shelf (J24 for 1796 test systems, J30 for 1799 test systems). Unscanned pins must be re-wired for each different type of measurement while scanned pins are switched automatically through the use of the scanner language described in para. F2.2.3. For example, the three typical types of resistance measurements shown in Figure F-9 can be connected in the following manner:

**For Two Terminal Connections**

Scanner connections for two terminal unguarded or guarded measurements require the use of the keyword **GUARD**. This keyword must be inserted prior to the AT construct. If **GUARD** is not included, connections for four terminal measurements **will** be made **AND** the automatic channel assignments for both **are** different.

Two terminal **guarded** connections **require** the use of all four channels (SA, SB, MA, MB). The Guard lead (G) is assigned to the MA channel while the Guard Sense lead (GS) is assigned to the MB channel. In addition, four pins should be specified in the pin list. For example, the statement

```
SET RMU GUARD AT(1:2:3:4);
```

will make the following automatic two terminal **guarded** connections:

|          |         |         |     |      |
|----------|---------|---------|-----|------|
| Pin:     | 1       | 2       | 3   | 4    |
| Channel: | SA      | SB      | MA  | MB   |
| RMU:     | (P+,I+) | (I-,P-) | (G) | (GS) |

Two terminal **unguarded** connections also require the use of the keyword **GUARD** and are made by listing pin assignments for the first two channels only. The last two channels have no pin connections and therefore must be cleared. Zeros may be entered into pin lists to clear the unwanted channels. For example, the statement

```
SET RMU GUARD AT(1:2:0:0);
```

will make the following two terminal **unguarded** connections:

|          |         |         |
|----------|---------|---------|
| Pin:     | 1       | 2       |
| Channel: | SA      | SB      |
| RMU:     | (P+,I+) | (I-,P-) |

If the channels that are not to be used have already been cleared previously, it will only be necessary to list the two pin connections required.

```
SET RMU GUARD AT(1:2);
```

Note, however, that this SET statement assumes that the last two channels **have** been cleared previously (perhaps by use of a **CLEAR RMU** statement).

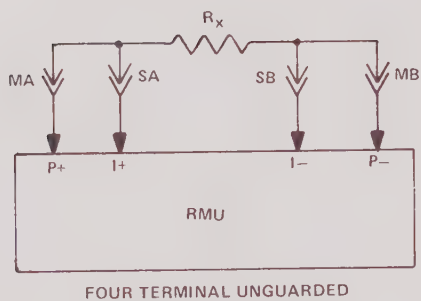
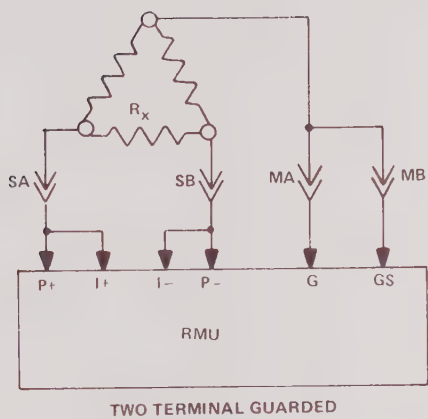
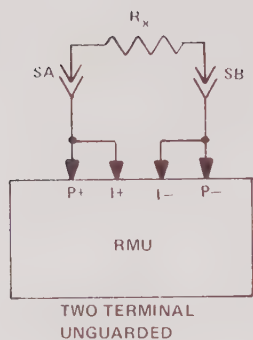


Figure F-9. Typical RMU Measurement Connections (Simplified).

Terminals may be assigned to the digital shelf through the Hybrid pin scanner. For example, the statement

```
SET RMU GUARD AT(H=1:H=2:0:0);
```

would assign the SA channel to D/S pin 1 and the SB channel to D/S pin 2.

The explicit scanner construction listed in Para. F2.2.2 may also be used to actually describe the connection assignments. For example, the statements:

```
CLEAR RMU;
(1) SET RMU GUARD AT(VP, IP @ 1:VM, IM @ 2);
 or
(2) SET RMU GUARD AT(1:2);
```

are equivalent, however, the explicit form at (1) completely describes the connections made. Table F-3 lists the various mnemonic and terminal names available for use. Care must be taken, however, not to mix channel names and mnemonics (or terminal names) in the same scanner device specification. For example, the statement

```
SET RMU GUARD AT(VP, IP @ 1: SB @ 2);
 { } { }
 Mnemonic Channel
 Names Name
```

is **illegal** because it mixes both mnemonic names and channel names in the same specification.

### For Four Terminal Connections

Scanner connections for four terminal unguarded measurements must be made without the keyword GUARD. For example, the statement

```
SET RMU AT(1:2);
```

will automatically make the following four terminal connections:

| Pin | Channel | Function |
|-----|---------|----------|
| 1   | MA      | P+       |
| 1   | SA      | I+       |
| 2   | MB      | P—       |
| 2   | SB      | I—       |

Note that automatic **two** pin assignments (for four channels) can be made through the analog scanner only (**not** the hybrid scanner) because the analog scanner consists of separate SOURCE and separate MEAS pins while the hybrid scanner does not. Four pin assignments are normally required for specific four terminal connections. For example, the statement

```
SET RMU AT(4:6:9:11);
```

will assign pins through the analog scanner as follows:

|          |    |    |    |    |
|----------|----|----|----|----|
| Pin:     | 4  | 6  | 9  | 11 |
| Channel: | MA | MB | SA | SB |
| RMU:     | P+ | P— | I+ | I— |

Note that pins 4 and 9 together with pins 6 and 11 must be Kelvin-connected together at the shelf device adaptor for this type of measurement (see Figure F-9).

The statement

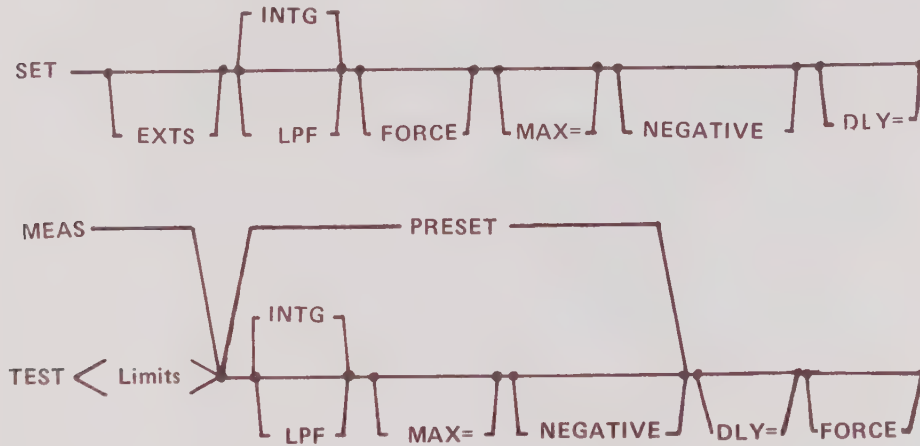
```
SET RMU AT(H=1:H=2:H=3:H=4);
```

will assign pins through the hybrid scanner as follows:

|          |    |    |    |    |
|----------|----|----|----|----|
| D/S Pin: | 1  | 2  | 3  | 4  |
| Channel: | MA | MB | SA | SB |
| RMU:     | P+ | P- | I+ | I- |

Note again, that the P+, I+, and P-, I- pins must be connected together at the shelf device adaptor for this type of measurement.

The following multimeter keywords listed in Table F-5 may be used with RMU SET, MEAS, or TEST statements:



Follow each SET, MEAS, or TEST path from left (start) to right (end) through the diagram.

Where branches occur, any **one** of the paths may be taken. Note, however, that it is **not** possible to include the keywords INTG and LPF in the same SET, TEST, or MEAS statement. Any of the keywords in a SET statement may be invoked by the keyword PRESET in a MEAS or TEST statement that follows. Note, however, that it is not possible to include INTG, LPF, MAX=, or NEGATIVE in a MEAS or TEST statement when PRESET is invoked.

The keywords shown may be used with any of the measurement connections (two terminal, four terminal, etc.). For example, the statement

```
MEAS RMU AT(1:2)MAX=1.5K;
```

will set the RMU up for a four terminal unguarded measurement at the analog shelf pins. The maximum resistance value to be measured is 1.5K ohms. Note that the RNG= keyword available for use with other measurement modes (ACV, DCV, DCI) **cannot** be used with the RMU (always specify MAX=).

TEST statements allow measured RMU results to be stored in the RMU register and then compared against the limits <Limits> given in the statement. For example, the statement

```
200: TEST RMU GUARD AT(H=1:H=2:Ø:Ø) (HI=10K LO=9.9K) [250] MAX=12K;
```

will make the comparison, and then branch to label 250 if the test conditions shown (HI=10K LO=9.9K) are not met.

The keyword NEGATIVE is used to reverse the polarity of RMU current or voltage forcing so that a second measurement can be made to cancel out any thermal emf's caused by relay closures. For example, the statement

```
MEAS RMU AT(1:2)MAX=2K NEGATIVE;
```

will invoke the keyword NEGATIVE.

This keyword results in the cancellation of any effects from thermal emf's. The cancellation results from first sourcing then sinking current through the unknown resistance. The two readings are then averaged (one with NEGATIVE and one without).

Any RMU measurements in excess of 200K ohms should also include either INTG or LPF to reduce noise pick-up.

The following commands can be used to clear RMU connections:

| Command   | Function                                                           |
|-----------|--------------------------------------------------------------------|
| CLEAR RMU | Clears (disconnects) RMU                                           |
| CLEAR ALL | Clears <b>all</b> devices and connections                          |
| (Ø:Ø)     | A Zero in a pin list clears the channel attached to that pin only. |

## High Compliance Current Measurements (DCVIF)

The optional DCVIF mode of the precision multimeter is entered whenever DCVIF is specified in SET, MEAS, or TEST statements. The high compliance current source (DCVIF) is useful for non-linear network measurements, since the voltage compliance of the current source is the full scale range of the instrument and is not limited by the 2.56V swing of the FMU function. It can be used to measure Zener Diode voltages or forward conductivity of diodes. Measurements are made by forcing a known current through a UUT circuit and measuring the voltage.

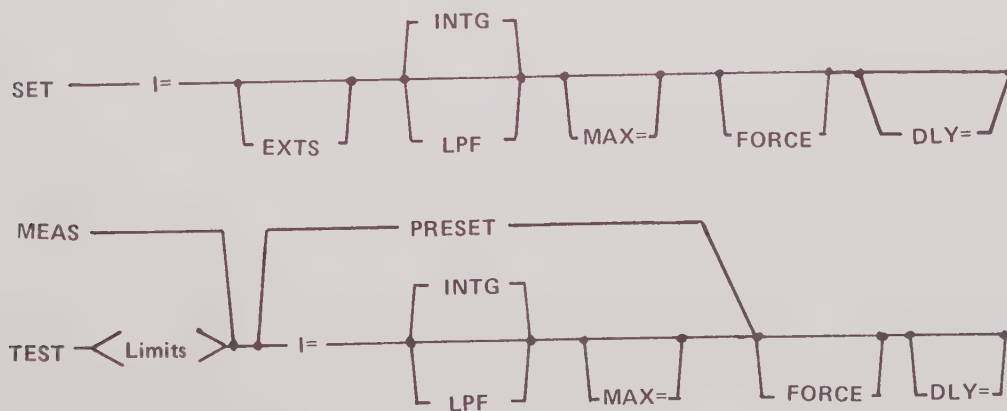
### WARNING

As with any open-circuited current source, the DCVIF can go to its full voltage compliance when unloaded. Open circuited shelf pins that are connected through the scanner to DCVIF can develop voltages as high as 160 V during test operations. Always make sure that shelf pins are connected to a UUT load prior to use. Devices tested with this function should also be shielded from an operator.

DCVIF connections to the circuit under test are two terminal connections and are **always** to the same two channels (SA, SB) used for two-terminal unguarded resistance measurements (see Figure F-9). The MA and MB channels are **not** used with DCVIF.

DCVIF also has a **non-scanned** guard circuit which may be employed. Pins 41(GSENSE) and 42(GUARD) at J24 on the analog shelf may be used for this purpose.

The following multimeter keywords listed in Table F-5 may be used with DCVIF SET, MEAS, or TEST statements:



Follow each SET, MEAS, or TEST path from left (start) to right (end) through the diagram.

Where branches occur, any **one** of the paths may be taken. Note, however, that it is not possible to include the keyword LPF in any of the statements if the keyword INTG is used. Note also, that the keyword I= must either be included in MEAS/TEST statements that do not invoke PRESET or included in a SET statement that is invoked by the keyword PRESET in a MEAS or TEST statement that follows. Any of the keywords in a SET statement may be invoked by the keyword PRESET in a MEAS or TEST statement that follows. Note, however, that it is not possible to include I=, INTG, LPF, or MAX= in a MEAS or TEST statement when PRESET is invoked.

The keyword I=n is a floating point value (or register) that contains a forcing current value (n). This value selects the DCVIF forcing current to be used and **must** be within  $\pm 5\%$  of one of the following values or a software error will occur:

- 2 mA
- 1 mA
- 100  $\mu$ A
- 10  $\mu$ A

For example, the statement

```
MEAS DCVIF AT(4:5)I=10U;
```

will connect the SA channel (P+, I+) to scanned SOURCE pin 4 and the SB channel (I-, P-) to scanned SOURCE pin 5 on the analog shelf. A 10  $\mu$ A forcing current will be selected.

The following commands can be used to clear DCVIF connections:

| Command     | Function                           |
|-------------|------------------------------------|
| CLEAR DCVIF | Clears (disconnects) DCVIF         |
| CLEAR ALL   | Clears all devices and connections |

Automatic channel/pin assignments for both the RMU and optional DCVIF when using the scanner language AT construct are as follows:

| Device | Type Connection      | Keyword | AT(            | :              | :          | :          | ); |
|--------|----------------------|---------|----------------|----------------|------------|------------|----|
| RMU    | 4-Terminal Kelvin    | None    | MA<br>(V+)     | MB<br>(V-)     | SA<br>(I+) | SB<br>(I-) |    |
|        | 2-Terminal Guarded   | GUARD   | SA<br>(V+, I+) | SB<br>(V-, I-) | MA<br>(G)  | MB<br>(GS) |    |
|        | 2-Terminal Unguarded | GUARD   | SA<br>(V+, I+) | SB<br>(V-, I-) | -          | -          |    |
| DCVIF  | 2-Terminal Unguarded | None    | SA<br>(V+, I+) | SB<br>(V-, I-) | -          | -          |    |

F2.3.5 Analog Probes (SET PROBE)

Up to two analog probes may be connected through jacks J1 (PROBE 1) and/or J2 (PROBE 2), at the front of the analog probe scanner box, to the probe scanner relays (see Figure F-8). The probe scanner relays interconnect the probes either to the MUX channels (MA, MB, SA, SB) or to a spare connection of J3 at the rear of the analog probe scanner box, or to the High and Low non-Kelvin connected outputs of UPS-2. Scanner connections may **not** be made to the MUX channels **and** to the SPARE and/or UPS-2 terminals at the same time. The SPARE and UPS-2 terminals are therefore dedicated terminals for use of the analog probes only and cannot be connected through to the MUX MA, MB, SA, SB channels at any time.

Both a general purpose analog probe (TYPE=1) and an analog diagnostics probe (TYPE=2) are currently available for connection to the probe scanner. Both hand-held probes are interconnected to a single measure channel and a single source channel through four sets of relay contacts in each probe. The relay contacts are controlled by test program SET PROBE statements which also set-up the probe scanner relays so that either a single MUX source and/or single MUX measure channel or the SPARE and UPS-2 terminals may be connected to each probe. Once the proper scanner connections

have been established through use of the SET PROBE statement, a second SET, MEAS, or TEST statement is normally required to set-up and activate the device (RMU, DCV, etc.) connected to the probe.

The general format of the SET PROBE statement is:

SET PROBE-n TYPE=m DIRMSR DIRSRC AT(MA:SA);

Where:

SET PROBE-n                      Sets-up the scanner relays for the connections specified by the AT( : ) construct, and connects the correct probe jack (J1 and/or J2) specified by -n. Probes connect to J1 (PROBE 1) and/or J2 (PROBE 2); -n must therefore be 1 or 2. For example,

SET PROBE-2;

connects PROBE 2, while

SET PROBE-1;

connects PROBE 1.

TYPE=m                              Specifies the type of probe to be used. There are currently two types of probes available and m may therefore be 1 or 2. Up to 15 different types can be supported by the software. If the entire TYPE=m is omitted, TYPE=1 is **assumed**. For example,

SET PROBE-2 TYPE=2;

connects the TYPE=2 analog diagnostics probe to J2 (PROBE 2), while

SET PROBE-2; or  
SET PROBE-2 TYPE=1;

connects the TYPE=1 general purpose probe to J2 (PROBE 2). The TYPE=1 probe may also be connected to J1 (PROBE 1) through use of the following default construct:

SET PROBE-1;

The TYPE=m construct **must** precede any relay keywords, if used.

DIRMSR DIRSRC                      These keywords (which must always be inserted before the AT construct) determine the number of probe relay contacts that are to close. In this example, the TYPE=1 direct measure (DIRMSR) contacts and direct source (DIRSRC) contacts are closed. At least one space must be inserted between each keyword listed. The following keyword combinations are possible (see Figure F-8):

#### TYPE=1 General Purpose Probe

|        |        |                |
|--------|--------|----------------|
| DIRSRC | (or B) | Direct Source  |
| DIRMSR | (or C) | Direct Measure |
| RCSRC  | (or A) | RC Source      |
| CMSR   | (or D) | C Measure      |

DIRSRC and DIRMSR contacts allow devices to be connected directly to the probe tip while CMSR and RCSRC can be used when Capacitance coupling to a Measure circuit (perhaps ACV) or Resistance-Capacitance coupling of a Source output is required. All four contacts can be closed individually or in any combination. The letters A through D shown in parenthesis may also be used in place of the longer keywords, if desired.

## CAUTION

Be careful when closing the probe relay contacts. Any combination of contacts can be closed making it possible to connect source and measure channels together at the probe tip, short-out C or RC coupling that may be required, etc. (see Figure F-8).

### TYPE=2 Analog Diagnostics Probe

|   |                      |
|---|----------------------|
| A | 20K Source           |
| B | 2K Source            |
| C | 180 Ohm Measure      |
| D | 0.1 $\mu$ fd Measure |

All combinations of contact closures are possible. For example, the contacts may be closed individually, two at a time, three at a time or four at a time. This is a special purpose probe designed for use with GR's analog diagnostics software, and is described further in the manual supplied with the optional analog diagnostics software.

AT(MA:SA);

The AT construct defines the MUX channels (MA, MB, SA, SB) **or** SPARE and UPS-2 terminals (T-25, T-26, and T-27) that are to be scanned to the probes. Each probe can accept only **one** measure and **one** source channel. The maximum number of channels that can be scanned per probe are therefore two (one measure and one source).

The SPARE and UPS-2 terminals are scanned through use of the following probe terminal numbers:

| Terminal Number | Connects              |
|-----------------|-----------------------|
| T-25            | HI (+) for PS-2 (UPS) |
| T-26            | LO (—) for PS-2 (UPS) |
| T-27            | SPARE 1 (J3, Pin 9)   |

Note that the UPS-2 terminals (T-25 and T-26) connect only to the source channels (not Measure) while the SPARE terminal (T-27) can be connected to either a source or a measure channel. M and S tokens may be added to the terminal numbers to specify connections to Source or Measure channels. If the tokens are not used, Source is assumed. For example, the statements

```
SET PROBE-1 AT(T-25:T-27M);
or
SET PROBE-1 TYPE=1 AT(T-25S:T-27M);
```

are identical and will both connect the TYPE=1 probe through J1 (PROBE 1) to the SPARE terminal at J3 and to the HI side of UPS-2. Note that it was not necessary to list the token S after T-25 as S is assumed for all terminals without tokens. Note also that it is not possible to use the M token for UPS-2 as the power supply must always be assigned to Source channels. It is also not possible to connect more than one UPS-2 terminal to the same source channel. Only one terminal can be connected to any one source or measure channel at one time.

The MUX channels (MA, MB, SA, SB) are connected to the probe by specifying one measure and/or one source channel per probe in the AT construct. Any SPARE or UPS-2 terminals that are connected to the source or measure channel specified will automatically be cleared when the MUX channel is connected. For example, the statement

```
SET PROBE-1 DIRMSR AT(MA);
```

would connect the MUX MA channel through J1 (PROBE 1) to the TYPE=1 probe. The Direct Measure (DIRMSR) contacts in the probe would be closed. The statement

```
SET PROBE-1 DIRMSR DIRSRC AT(MA:SA);
```

would connect the MUX MA channel through DIRMSR and the SA channel through DIRSRC to the TYPE=1 probe tip. The statement

```
SET PROBE-2 TYPE=2 A D AT(MB:SB);
```

would connect both the MUX IMB and SB channels through J2 (PROBE 2) to the TYPE=2 probe. The A and D TYPE=2 probe contacts would be closed for this operation.

To prevent measurements at multiple UUT points, Hybrid and Analog scanner shelf pins should **not** be connected to the probe scanner during a probe measurement. One method of preventing this from happening is to clear the measure or source device being used prior to connecting it to the probe scanner (through use of CLEAR DCV, CLEAR RMU, etc.). Another method involves the use of a zero in the AT construct for those pins that should be cleared. For example, the statements

- (1) MEAS DCV AT(5:10)MAX=5;
- (2) SET PROBE-1 DIRMSR AT(MA);
- (3) MEAS DCV AT(HI @ 0:LO @ \*)MAX=5;

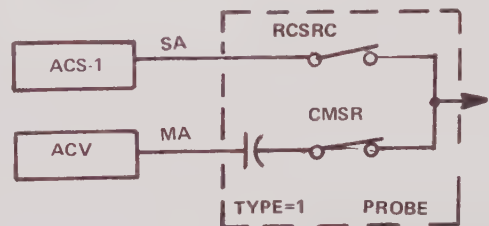
would (1) connect the DCV to pins 5 (HI) and 10 (LO) on the analog shelf for a standard DCV measurement. Next, a SET PROBE statement (2) sets-up the TYPE=1 probe for a direct measurement (DIRMSR) on the MUX MA channel. Finally, a probe DCV measurement (3) is performed with the DCV HI side connected through the MA channel to the probe and the DCV LO side connected through analog shelf pin 10 to the UUT. The zero at (3) clears any analog or hybrid shelf pins that may be connected to the MA channel (in this case, pin 5). The asterisk (\*) indicates “no-change” in pin status (use pin 10 again).

The following examples illustrate some of the methods that may be used to interconnect the probes for use:

## ACV Measurements

The following statements set-up the TYPE=1 probe through J1 (PROBE 1), connect the ACV and ACS to the probe tip, and finally perform an ACV measurement:

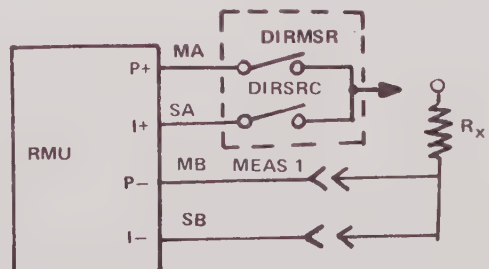
```
SET PROBE-1 RCSRC CMSR AT(MA:SA);
.
.
SET ACS F=100 V=3 AT(SA);
.
.
MEAS ACV AT(MA)MAX=5;
```



### Four-Terminal Kelvin Resistance Measurements

The following statements set-up the TYPE=1 probe through J2 (PROBE 2), connect the RMU for 4-terminal Kelvin measurements and make the measurements:

```
SET PROBE-2 DIRSRC DIRMSR AT(MA:SA);
.
.
CLEAR RMU;
MEAS RMU AT(*:1:*:1)MAX=1.5K;
```



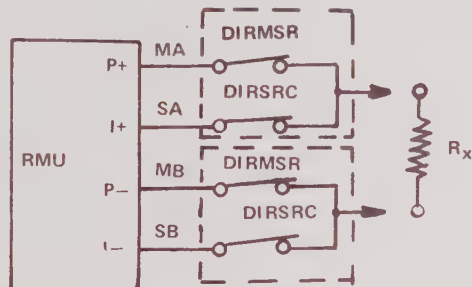
Note that the use of an asterisk (\*) assumes that prior RMU channel pin connections to the shelf (if any) have been cleared. The statement

```
MEAS RMU AT(MA @ 0:MB @ 1:SA @ 0:SB @ 1)MAX=1.5K;
```

could also have been used without a prior CLEAR RMU statement to clear any analog or hybrid pins that were connected to channels that the probe would use.

If two TYPE=1 probes were available, then the following configuration could be used to set-up both probes for a 4-terminal Kelvin measurement:

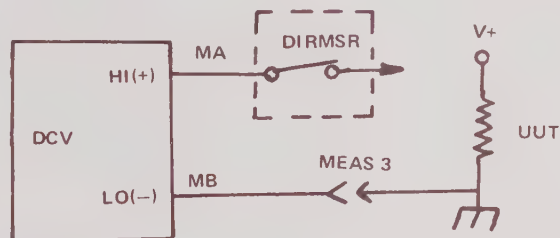
```
SET PROBE-1 DIRSRC DIRMSR AT(MA:SA);
SET PROBE-2 DIRSRC DIRMSR AT(MB:SB);
.
.
CLEAR RMU;
MEAS RMU MAX=1.5K;
```



### DCV Measurements

The following statements set-up the TYPE=1 probe through J1 (PROBE 1), connect the DCV HI side to the probe and the DCV LO side to the analog shelf, and finally make the DCV measurement:

```
SET PROBE-1 DIRMSR AT(MA);
.
.
CLEAR DCV;
MEAS DCV AT(*:3)MAX=5;
```



### SPARE and UPS-2 Measurements

The following statement sets up the SPARE terminal at the rear of the probe scanner box for use at J1 (PROBE 1):

```
SET PROBE-1 DIRMSR AT(T-27M);
```

The following statement sets-up the UPS-2 power supply for use. In this example, the HI side only is probed. The Low side can be scanned through the power supply scanner to a shelf pin. If two probes are available, the HI side can be scanned to one probe and the LO side to the second probe as the following examples indicate:

```
SET PROBE-1 DIRSRC AT(T-26);
or
SET PROBE-1 DIRSRC AT(T-25);
SET PROBE-2 DIRSRC AT(T-26);
```

### CLEAR Commands

The following commands can be used to clear the probes at any time:

| Command       | Function                                  |
|---------------|-------------------------------------------|
| CLEAR PROBE   | Clears <b>all</b> probes.                 |
| CLEAR PROBE-n | Clears the probe specified only (1 or 2). |

It should also be noted that the four relay contacts for each probe are automatically opened (cleared) prior to setting-up the required closures for each SET PROBE statement.

F2.4 UUT Power Supplies

A 1796 Test System may have one, two, or three UPS;TPS boxes installed. Each Box may contain either one Triple Power Supply or from one to three Universal Power Supplies. Alternatively, a fixed user supply can be installed in the first slot of UPS/TPS Box #1. If installed, the fixed user supply is always referenced as PS-1.

The possible UUT power supply configurations that a 1796 test system can support are shown in Figure F-10. The physical constraints which each system must satisfy are: first, that a fixed user supply, if installed, is always located in Box #1, slot 1; second, a Triple Power Supply, if installed, requires an entire Box; and third, an 80V Universal Power Supply, if installed, requires two adjacent slots in the same Box. There are no constraints on mounting 10V, 20V, or 40V UPSs.

UPS programming on a 1796 is identical to that for a 1792-C/D, which is discussed in Paragraph 3.2.7.3. Programming statements for the fixed and Triple power supplies are similar to those given in Paragraph 3.2.7.1 for a 1792-A, but with overriding considerations described in the following paragraphs.

Power Supply Scanning

Installation of the Power Supply Scanning option enables the test program to establish connections between UPSs and UUT pins, as described in Paragraph 3.2.7.3. In a given system, five power supply terminals can be scanned to as many as 36 UUT pins. Seven different combinations of terminals are allowed; each system is factory-wired for one of the seven. These combinations, or "scanning codes", are described in the System Operation manual, Section 2.

Power supply scanner connections are specified using a pin-list preceded by AT, as described in Para. 3.2.4; the primary-pin-specification is the high (output) terminal of the indicated power supply, and the secondary-pin-specification is the low (return) terminal. An attempt to connect a terminal which is not scanned in a particular system will result in an error, when the statement is executed on that system.

As an example, consider the statement:

```
9: SET PS-5 AT(5:12) V=30 I=150M;
```

The resulting connections will be between the high terminal of PS-5 and UUT pin 5, and between the low terminal of PS=5 and UUT pin 12.

Further considerations on the programming of power supply scanners are given in Para. 4.2.

Fixed user supplies and Triple Power Supplies can not be scanned, because their current capability exceeds that of the scanner system. If a test statement containing a scanner specification refers to a power supply which is either fixed or part of a TPS, an interpreter error will result.

Fixed User Power Supply

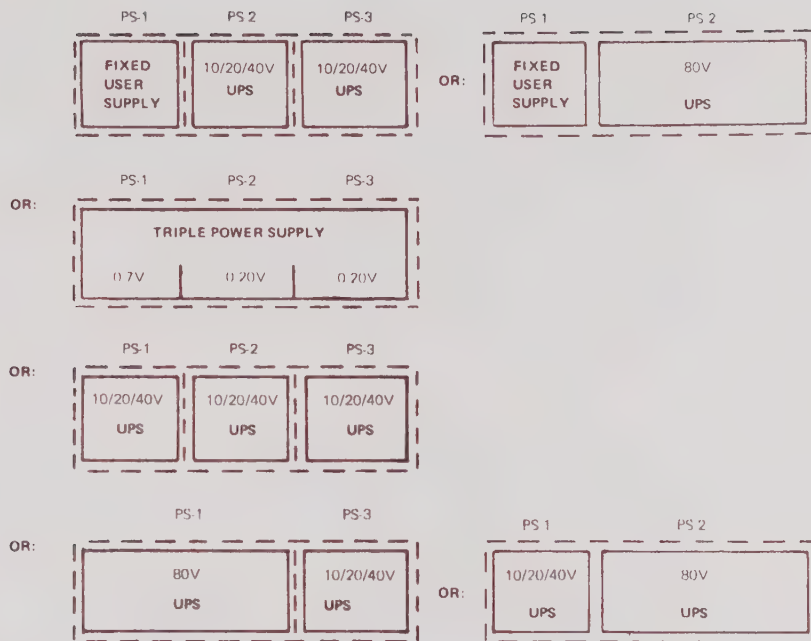
The fixed user supply provides three voltage outputs rated at 10 amperes maximum and, when installed, must be referenced as PS-1. The three nominal voltage outputs are 4.75 volts (low), 5.00 volts (normal), and 5.25 volts (high). The low, normal, and high outputs respectively correspond to the PS-1A=, PS=1B=, PS-1C= ranges of the SETUP Mode Power Supply Configuration page. Normally, for the fixed user power supply, this page would be setup as follows: (note that the hundreths digit must be 5 or 0).

```
PS-1: 1A=4.65, 4.85 1B=4.90, 5.10 1C=5.15, 5.35
```

Then, the following SET PS-1 statement would select the low range output (4.75 volts) of the fixed user power supply because the voltage specified (V=4.7) is within the range setup for 1A above.

```
100: SET PS-1 V=4.7 I=10 DLY=50 [80];
```

UPS/TPS Box # 1:



UPS/TPS Box # 2/3:

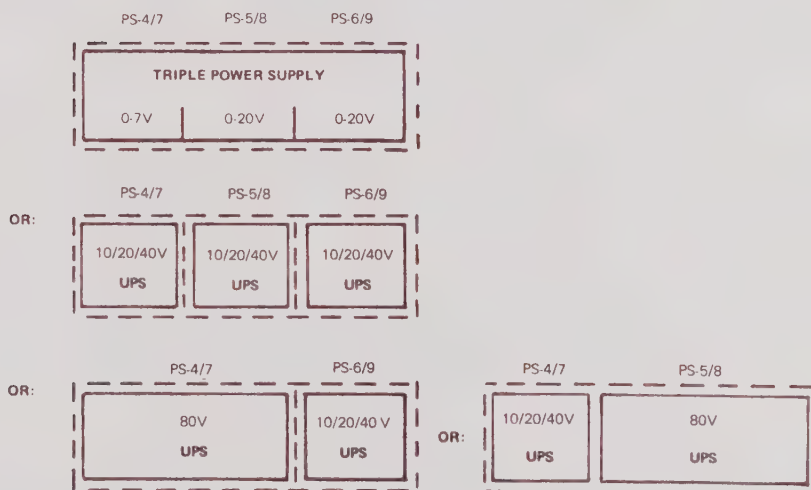


Figure F-10. 1796 UUT Power Supply Configurations.

As shown in the above statement, I=10 must always be specified when programming the fixed user power supply. The statement label (100:), the DLY=50 parameter, and the error branch (80), are used as described in the manual.

#### NOTE

The V= and VO= options of DIAGNOSIS Mode also affect the selection of the power supply output. That is; when V=HIGH or V=LOW is specified on the Operator Controls page of DIAGNOSES Mode and the VO= option of the Select Miscellaneous Options page is specified as a numeric percentage, the computer accordingly calculates an adjusted voltage for the V= specification in the SET PS-1 statement. This calculated adjusted voltage is then compared to the ranges specified on the Configuration page of the SETUP Mode to select the power supply output.

### Triple Power Supply

The triple power supply consists of three independent sections: PS-1 (or PS-4 or-7), PS-2 (or PS-5 or -8), and PS-3 (or PS-6 or -9). Each section provides three voltage outputs. The actual voltages at the outputs are preset by hardware (resistor plugs). Each of the three voltage outputs of the first section (PS-1, 4 or 7 can be preset to any voltage between 0.5 volts and 7.5 volts at 5 amperes maximum (nominal 5-volt plug supplied). Each of the three voltage outputs from the remaining two sections (PS-2 and PS-3, PS-5 and PS-6 or PS-8 and PS-9) can be preset to any voltage between 0.5 volts and 20 volts at 1.5 amperes maximum (nominal 15-volt plugs supplied). The three voltage outputs of each section are viewed as output A, output B, and output C. Voltage ranges corresponding to the three outputs from each section are specified on the Power Supply Configuration page of SETUP Mode. Assume this page is setup as follows:

|       |                 |                 |                 |
|-------|-----------------|-----------------|-----------------|
| PS-4: | 4A=4.65, 4.85   | 4B=4.90, 5.10   | 4C=5.15, 5.35   |
| PS-5: | 5A=13.40, 13.70 | 5B=14.90, 15.10 | 5C=16.40, 16.60 |
| PS-6: | 6A=13.40, 13.70 | 6B=14.90, 15.10 | 6C=16.40, 16.60 |

Then, the following command statement selects output A of the first section (PS-4) because the value of the V= specification falls within the range specified for the 4A=parameter:

```
110: SET PS-4 V=4.7 I=5 DLY=50 [80];
```

Similarly, the following command statements select outputs B and C of the second (PS-5) and third (PS-6) sections respectively:

```
120: SET PS-5 V=15 I=1.5 DLY=50 [80];
125: SET PS-6 V=16.40 I=1.5 DLY=50 [80];
```

Note that the V= specification of the command statement only selects the output of the power supply; the actual voltage at that output is preset by the resistor plug and does not necessarily reflect the V=value. The actual (nominal) voltages at outputs A, B, and C with the factory-supplied 5-volt plug are 4.75, 5.0, and 5.22 ( $\pm 5\%$ ) respectively; and with the 15-volt plug are 13.66, 15.04, and 16.55 ( $\pm 10\%$ ).

The I= specification in the command statement must be I=5 when setting PS-1, PS-4 or PS-7 and must be I=1.5 when setting the other two sections of the triple power supply. The other specifications (statement label, error branch, and DLY= specification) are programmed as described in the manual.

#### NOTE

See preceding note (under Fixed User Power Supply).

## **F2.5 PROGRAMMING NON-SCANNED DEVICES**

### **F2.5.1 Time Measurement Unit (TMU)**

The Time Measurement Unit (TMU) is a device that is capable of making high-precision time interval measurements such as rise or fall times, pulse durations, and pulse separations. TMU measurements are controlled by START and STOP trigger pulses which are applied through pins on the digital shelf (refer to Section 2 of the System Operation Manual). The TMU is programmed using the procedures described in Para. 3.2.17.

### **F2.5.2 Relay Drivers (RLY)**

Relay drivers are open collector transistor circuits. The non-scanned relay driver outputs are applied to pins on the analog shelf where they may be used to energize relay coils or indicator lamps in the device adaptor or UUT (refer to Section 2 of the System Operation Manual). The relay drivers (RLY) may be programmed using the procedures described in Para. 3.2.9.

### **F2.5.3 Sync Pulse Generation (SYNC)**

Positive and Negative going TTL-compatible SYNC pulses can be generated under program control. The pulses are applied to shelf pins and BNC connectors on the digital shelf (refer to Section 2 of the System Operation Manual). SYNC pulses may be programmed using the procedures described in Para. 3.2.8.





1792A Test System, 3-17  
 1792B Test System, 3-20  
 1792C Test System, 3-22  
 1792D Test System, 3-22  
 1795/1799 Test System, D-1  
 1796/1799 Test System, F-1  
 1797 Test System,  
 1799 Hardware Start, 3-34  
 #, 3-3, 3-13, 3-57  
 \$, 3-5, 3-57  
 %, 3-48, 3-50

- A -

A (1796 Probe), F-48  
 Abort Exit, B-7  
 ABSOLUTE Command, 3-41  
 AC Source (ACS) 1-4, 3-29, F17, F-33  
 AC Voltmeter (ACV), 1-4, 3-28, F-18,

F-33, F-35

AC-DC Converter (DCV), 1-4, 3-27, F-18,

F-33, F-35

ACS, 1-4, 3-29, (1796/99) F-18, F-33  
 ACV, 1-4, 2-8, 3-28 (1796/99) F-18,  
 F-33, F-35

ACV Keywords (1796/99), F-35, F-40  
 ACV Probe Measurements (1796/99), F-51  
 ACV Register, 3-28  
 ADD Command, 3-40  
 ADIR (FMU), 2-5, 3-30  
 All But, 3-3, 3-13, 3-57  
 Analog Pin Tokens (1796), F-30  
 Analog Probes (1796), F-48  
 Analog Scanner (1796), F-15, F-20  
 Analog Test Programming (1792), 4-3  
 AR, 2-8  
 Arithmetic Commands, 3-1, 3-40, A-4  
 Arithmetic Register (AR), 2-8, 3-40  
 ASCII Bus (IEC), 1-4, E-1  
 ASCII Character Set, B-11, B-16  
 ASSIGN LGC, 3-5, 3-6  
 ASSIGN PNAME, 3-2, 3-3, 3-53  
 Asterisk (\*), 3-3, F-26  
 AT Prefix, 3-2, 3-3, F-20  
 ATAN Command, 3-42  
 ATEST Command, A-4

- B -

B (1796 Probe), F-49  
 BDIR (FMU), 2-5, 3-30  
 BEGIN CAPS Command, 3-4, C-1  
 BEGIN FAST, (1796) F-4, F-6  
 BEGIN RECORD, 3-4, C-6  
 Beta, C-8  
 BINFLD, B-8  
 Blinking Text On Screen, 3-51  
 BLOCK, 3-43  
 Branch Addresses, 2-8  
 BRANCH Command, 3-43, C-2  
 BRANCH Exit, B-6  
 Breakpoint, B-14  
 Burst Mode (1796), F-4  
 Bus (IEC), 1-4, E-2, E-3

- C -

C (1796 Probe), F-49  
 CALL, 3-44, C-2  
 CAPS, C-1  
 CAPS Monitor, 1-2  
 CAPS Multiple Sections, C-3  
 CAPS Recording Section, 3-4, C-1  
 CAPS Related Techniques and

Commands, C-1  
 CASE UNTIL (TGL), C-8  
 CCOMP, 3-47, C-2  
 CHAR, 2-8  
 CHAR Register, 3-47, 3-51  
 Character Comparison, 3-47  
 Character Set, 2-2  
 CHECK, 2-4  
 CLEAR ACS, 3-29  
 CLEAR ACV, 3-29  
 CLEAR ALL, 3-2, 3-4, 3-8, 3-27, 3-28,  
 3-38, 3-47, C-1, F-29, F-42  
 CLEAR BUS, E-2  
 CLEAR DCI, 3-28, (1796/99) F-42  
 CLEAR DCV, 3-27, (1796/99) F-42  
 CLEAR LGC, 3-5, 3-7  
 CLEAR MASCAN, 3-37, F-29  
 CLEAR MBSCAN, 3-37, F-29  
 CLEAR PmSCAN, 3-37  
 CLEAR RLY Command, 3-27  
 CLEAR SASCAN, 3-37, F-29  
 CLEAR SBSCAN, 3-37, F-29  
 CLEAR SCAN, 3-37, F-29  
 CLEAR XMEAS, 3-38, F-29  
 CLEAR XSOURCE, 3-38, F-29  
 CLK (FMU), 3-30  
 CLOSE, 3-47  
 CMSR (1796), F-49  
 Comments, 2-2, 3-15,  
 COMP, 3-41, C-2  
 CONTINUE, 2-8  
 CONTINUE Button, 3-46, B-7  
 CONTINUE ON ERROR Option, 3-19  
 Core Boundary, 3-43  
 COS, 3-42  
 COUNT, 2-8  
 COUNT Register, 3-31  
 CURFLD, B-4  
 Current Locked (IL) Mode, 3-22  
 Current Sensing Resistor (DCI), 3-28

- D -

D (1796 Probe), F-49  
 DA File, C-3  
 Data Terminal Characters, 3-48  
 Data-List Acquisition, B-7  
 DC (ACS), 3-29  
 DC Current Meter (DCI)  
 1792 Systems 3-28  
 1796/1799 Systems F-33, F-35  
 DCV Register 3-27  
 DC Voltmeter (DCV)  
 1792 Systems 3-27  
 1796/99 Systems F-33, F-35  
 DC-To-Digital Converter (DCV),  
 1-4, 3-27, 3-28, F-33, F-35  
 DCI, 1-4, 2-8, 3-28, (1796/99) F-35  
 DCI Register, 3-28  
 DCL, 2-5, E-3  
 DCV, 1-4, 2-8, (1796/99) F-35  
 DCV Probe Measurements (1796), F-52  
 DCV/DCI Keywords (1796/99), F-36  
 DCVIF (1796/99), 1-5, F-47  
 DCVIF AT Construct (1796/99), F-48  
 DCVIF Keywords (1796/99), F-35, F-47  
 DCVPRB, 3-39  
 Debugging Overlays or Subroutines, B-14  
 DEFINE Macro, 3-63  
 Defined-Symbols Table, B-1  
 Degree Conversion, 3-42  
 DELAY, 3-46  
 Delta, C-8  
 Device Power, 3-17, 4-1, C-6, D-2, F-53

Device Power Supplies

1792A Systems, 3-17  
1792B Systems, 3-20  
1792C/D Systems, 3-22  
1795 Systems, D-2  
1796 Systems, F-53  
DIAGNOSIS Mode, 1-2  
Diagnostics Probe (Type -2), F-50  
Difference Datat, DD-1  
Digital Test Unit, 1-2  
DIRMSR (1796), F-49, F-51  
DIRSRC (1796), F-49, F-51  
Disk Cartridges, 1-3  
Disk Memory, 1-3  
DIV, 3-41  
DLY, 3-46, F-35  
DLY=n (1796/99), F-34, F-35  
DO Loops, 3-46, C-3, F-9  
DONE, 3-46, F-9  
Driver Sensor Pin, 1-3  
Driver Strobe, 1-3  
Drivers and Sensors, 3-4, D-1  
DSKn, 1-3

- E -

EDIT Mode, 1-2  
Editor, 2-1  
END, 3-43  
ENDCASE, C-8  
ENDM Macro, 3-63  
ERRET, 2-4, 3-45,  
Error Exit, B-6  
Error Return, 2-4, 3-45  
ESCAPE, 3-46  
EXECute, 3-52, B-1, B-5  
Execution Order, 2-1, 3-12, 3-64  
EXP, 3-42  
Exponential Form, 2-2  
Extended Program Execution, 1-3  
External Test Equipment (1792), 3-38  
(1796/1799), F-26  
EXTS (1796/99), F-35, F-36  
EXIT (1796), F-7

- F -

F (ACS), 2-5, 3-29  
FAC, B-5, B-9, B-11  
FAIL, 2-8  
FRATIO, 2-8  
FAST Block Operation (1796), F-3, F-4  
FAST Blocks (1796), F-3  
FAST Block Timing (1796), F-4  
FAST Example, F-10  
FAST Instructions (1796), F-4  
FCDF, B-12  
FEXT, B-9  
FILE, 2-5  
FINISH CAPS, 3-4, C-1  
FINISH FAST, F-6  
FINISH RECORD, 3-4  
FIRST, 2-5  
Fixed User Power Supply (1796), F-53  
Floating Point Data Item, B-8  
Floating Point Processor, B-11  
FLPCLR, B-12  
FLPOPT, B-12  
FMU, 1-4, 3-30 (1796/99), F-32  
FORCE (1792), 3-27 (1796/99) F-35, F-37  
FORMAT, 3-48  
FORMAT EXP, 3-48  
FPCHO1, B-12  
FPCHI1, B-12  
FPFLD, B-11  
FRATIO, 2-3, 3-31  
FREQ, 2-8, 3-31  
Function Generator, 1-4, 3-29  
Functions Reserved Words, 2-4

- G -

GATE (ACS), 2-5, 3-29  
General Registers, 2-8, B-9, C-6  
GET, 2-6, E-3  
GETFPN, B-8  
GETINC, B-8  
GETTXP, B-8  
GR Test Systems, 1-1  
GRAY Code, 3-61  
GSENSE, 2-6  
GTL, 2-6, E-3  
Gray Macro, 3-61  
GUARD (1796/99), F-43, F-48

- H -

Hardware Start (1799), 3-34  
HDERRF, B-6  
High-Speed Reader, 1-3  
High-Speed Testing (1796), F-1  
Home Position, 3-48, 3-51  
Hybrid Device Testing (1795/99), D-1  
Hybrid Pin Assignments (1796), F-30  
Hybrid Scanner (1796/1799), F-15, F-22

- I -

I=n (1796/99), F-36, F-48  
IA, 3-2, 3-5  
IB, 3-2, 3-5  
IEC (ASCII) Bus, 1-4, E-1  
IEC Commands, E-2  
IGNORE LGC, 3-5, 3-7  
IH, 3-2, 3-5  
IL, 3-2, 3-5  
IL Mode, 3-22  
IM, 2-6, F-21  
Incremental LGC States, 3-12  
Input/Output Commands, 3-7  
Inter-Strobe Time (IST)  
1792A/B Systems, 3-8  
1792C/D Systems, 3-9  
1795/99 Systems, 3-10  
1796 Systems, F-1  
Inter-Test Time (ITT)  
1792A/B Systems, 3-8  
1792 C/D Systems, 3-9  
1795/99 Systems, 3-10  
1796 Systems, F-1  
INTG (1796/99), F-35, F-37  
INTP, B-9, B-11  
IOERR, B-6  
IOFLD, B-10  
IONPS, B-7  
IP, 2-6  
IST (1792, 1795/1799) 3-5, 3-8 (1796) F-1  
ITT (1792, 1795/1799) 3-5, 3-8 (1796) F-1

- L -

Label, 2-8, 3-1, 3-14  
LD, 3-40  
LGC, 1-3, 3-4  
Limits, 2-1  
Line Printer, 3-48  
LLO, 2-6, E-3  
LMTCHK=, 3-19, 3-25  
LOAD, 3-40  
Load Mode (1796) F-4, F-5  
Loader Map, B-5  
LOG, 3-42  
Logging File, 3-47, 3-48  
Logic Level Set, 1-3, 3-6, D-1  
Logic Set, 3-4, 3-12  
Logic Test Commands, 3-5  
Logic Testing (1795/1799), D-1  
Logic Testing (1796), F-1  
Logic Tests, 3-4  
LOOP, C-8  
LOOP Macro, 3-55  
Logic Probe (Pulse Catching), 1-5, 3-39  
LOWZ (1792) 3-28 (1796/99) F-35, F-40  
LPF, 2-6, F-35, F-37  
LPT, 3-48, B-10

- M -

MAJFLD, B-4  
 MASCAN (1792) 3-37 (1796) F-26  
 MAX, 2-6, F-35, F-38  
 MBSCAN (1792) 3-37, (1796) F-26  
 MCOMP, 3-7  
 MCOMP LGC, 3-5, C-2  
 MEAS, 3-2  
 MEAS ACV, (1792) 3-29 (1796/99) F-40  
 MEAS COUNT, 3-31  
 MEAS DCI, (1792) 3-28, (1796/99) F-40  
 MEAS DCV, (1792) 3-27, (1796/99) F-40  
 MEAS FREQ, 3-31  
 MEAS PSI, 3-25  
 MEAS PSV, 3-26  
 MEAS TIME, 3-31  
 Measurement Channel (1792) 3-37,  
 (1796) F-15  
 MLA, 2-6, E-3  
 MSA, 2-6, E-3  
 MTA, 2-6, E-3  
 MPY, 3-41  
 MULT, 3-41  
 Multimeter, 1-5  
 Multimeter Keywords (1796/99) F-35  
 Multimeter Range Table (1796/99), F-34  
 MUX Channel Connections (1796), F-21

- N -

NEGATIVE (1796/99), F-35, F-46  
 Non-Recording Section, C-6  
 Non-Scanned Devices (1796/99), F-56  
 NOP Command, 3-43  
 Normal-Speed Logic Testing (1796), F-1  
 Normal-Speed Timing (1796), F-2  
 NPEAK, (1796/99) F-35, F-41  
 Null Command, 3-5  
 Numerical Values, 2-2

- O -

OA, 3-2, 3-5  
 OB, 3-2, 3-5  
 Octal Data Item, B-7  
 ODT, B-14  
 OH, 3-2, 3-5  
 OL, 3-2, 3-5  
 OPEN WRITE Command, 3-47, 3-48  
 Operating Instructions Manual, 1-1, 1-2  
 Optimizing Logic Tests, 3-16  
 OVERFL, B-4  
 Overlay Exits, B-6  
 Overlays, 3-52, B-1, B-4  
 OVERLY.PA, B-1

- P -

PAL8 Language Routines, 3-52, B-1  
 Parameters Reserved Words, 2-5  
 Passive Test Programming, 4-3  
 PAUSE Command, 3-46  
 PAUSE Exit, B-7  
 PAUSER, B-7  
 Percentage Character, 3-48, 3-50  
 PGEN, 1-4, 3-33 (1796/99) F-32  
 Physical Ordering, 3-12  
 Pin Scanner System (1792) 3-36 (1796) F-15  
 PIN States, 3-13  
 PIN-List, 3-2, 3-3, 3-12  
 PmSCAN, 3-37  
 PNAME, 3-53  
 Port Numbers (1792) 3-36 (1796) F-21, F-23  
 Position List (IEC), E-1  
 Positioning Text on Data Terminal, 3-48

Power Supplies

1792A Systems 3-17  
 1792B Systems 3-20  
 1792C/D Systems 3-22  
 1795 Systems D-2  
 1796 Systems F-53  
 1799 Systems D-5  
 PPEAK (1796/99) F-35, F-41  
 Precision Multimeter (1796/99), 1-5,  
 F-18, F-33  
 PREPARE Mode (CAPS) C-1  
 PRESET (1796/99), F-35, F-38  
 PRINT, 2-6, B-10  
 Probe (Pulse-Catching), 1-5, 3-39  
 Probe Resistance Measurements  
 (1796), F-51  
 Probe Scanner (1796/99), F-20, F-48  
 Probe SPARE and UPS-2 Measurements (1796),  
 F-45  
 Probe Terminal Numbers (1796), F-19, F-45  
 Program Considerations, 4-1, B-1, F-10  
 Program Execution, 1-2  
 Programmed Delays, 3-46  
 Programming Definitions, 2-1  
 PS-n  
 1792A Systems 3-17  
 1792B Systems 3-20  
 1792C/D Systems 3-22  
 1795 Systems D-2  
 1796 Systems F-53  
 PmSCAN, 3-37, 3-38  
 PSI, 2-8  
 PSI Register, 3-25  
 Psuedo-Random Numbers, 3-61  
 PSV, 2-8  
 PSV Register, 3-26  
 PTP, 3-48, B-10  
 PTR, 3-51, B-10  
 pulse Generator (1792) 1-4, 3-33  
 (1796/99) F-32  
 Pulse-Catching Probe, 1-5, 3-39

- R -

Radian Conversion, 3-42  
 Random Macro, 3-61  
 RCSRC (1796), F-49  
 READ DEV=, 3-51  
 Reading Commands, 3-51  
 READX, B-20  
 REASSIGN LGC, 3-5, 3-6  
 RECORD Command, C-1, C-6  
 RECT, 2-7, 3-29  
 REG-1 Through REG-100, 2-9, 2-10, 3-40  
 REGFLD, B-9  
 Register Operations, 2-10  
 Registers, 2-9, 2-10, 3-40, C-6  
 Relay Drivers (1792), 3-27 (1796/99) F-56  
 RELPRO (1796), F-31  
 REPEAT UNTIL, C-8  
 REPEAT/LOOP Macros, 3-55, C-3  
 RESERVE Command, 3-52, B-1, B-5  
 Reserved Words, 2-3  
 Resistance Measurement Unit (1792) 1-4,  
 3-32 (1796/99) F-42  
 Result Mode (1796) F-4  
 RETURN Command, 3-44, 3-45, C-2  
 Ring Counter, 3-60  
 RLY (1792) 1-4, 3-27 (1796/99) F-56  
 RMU (1792) 3-32 (1796/99) F-42  
 RMU AT Construct (1796/99), F-43  
 RMU Connections (1796/99), F-44  
 RMU Four-Terminal Connections  
 (1796/99), F-45  
 RMU Keywords (1796/99) F-35, F-42-44  
 RMU Two Terminal Connections  
 (1796/99), F-43  
 RNG=n (1792) 3-27 (1796/99) F-36, F-41  
 Rotate Macro, 3-60  
 Rub-Out Key, 3-51

- S -

Sample Programs, 5-1  
 SASCAN (1792) 3-36 (1796/1799) F-26  
 SBIT (ON/OFF), E-9  
 SBSCAN (1792) 3-36 (1796/1799) F-26  
 Scanner AT Construct (1796), F-20  
 Scanner Language (1796), F-20  
 SCOMP, 2-5  
 SDC, 2-7, E-3  
 Self-Initializing Code, B-4  
 Sensor Strobe, 1-3  
 Sensor Threshold Voltage, 1-3  
 SEQUENCE Macro, 3-57  
 SET ACS (1792) 3-29 (1796/99) F-33  
 SET FAST (1796) F-12  
 SET FMU (1792) 3-30 (1796/99) F-32  
 SET LGC, 3-5, 3-6, 3-57, 3-61  
 SET MASCAN (1792) 3-37 (1796/99) F-30  
 SET MBSCAN (1792) 3-37 (1796/99) F-29  
 SET PmSCAN Command, 3-37  
 SET POWER LMTCHK= 3-19, 3-25  
 SET POWER ON/OFF 3-19, 3-21, 3-24  
 SET PROBE-n (1796), F-44  
 SET PS 3-17, 3-18  
 SET PULSE, 3-5, 3-53  
 SET RLY, 3-27  
 SET SYNC, 3-26  
 SET XMEAS (1792) 3-38 (1796) F-23  
 SET XSOURCE (1792) 3-38 (1796) F-23  
 Sharp Symbol (#), 3-3, 3-13, 3-57  
 SHELF (1796/99), F-32  
 Shelf Pins, 3-2  
 SHIFTO Macro, 3-59  
 SHIFTI Macro, 3-59  
 SIN, 3-42  
 Slash Marks, 2-2  
 SLOW (1792) 3-28 (1796/99) F-36, F-40  
 Sample Programs, 5-1  
 SMOVE, 2-5  
 Source Language, 2-1  
 Source Program, 2-1  
 SPD, 2-7, E-3  
 SPE, 2-7, E-3  
 Special Characters, 2-3, 3-48  
 SQRT, 3-41  
 SQUARE, 3-41  
 Square Root (SQRT) 3-41  
 SREG, E-1, E-8  
 STEST, 2-5  
 Start and Stop Shelf Inputs, 1-4  
 STORE, 3-40  
 STROBE, 2-7, 3-33  
 SUB, 3-40  
 Subroutines, B-1, B-5  
 Subroutines Exits, B-6  
 Symbolic Terminal Designations, 3-53  
 Symbols Table, B-1  
 SYNC Pulse Generation, 3-26  
 System Macro Commands, 3-57  
 System Manuals, 1-2  
 SYSTYP, B-2

- T -

TD 2-7, 3-33  
 TEC, 3-43, 3-46, B-10  
 Teletype, 1-3, 3-3, 3-48  
 Terminal Designation, 3-2, 3-53  
 Terminal (T-n) Names (1796) F-21, F-29  
 TEST ACV (1792) 3-28 (1796/99) F-40  
 TEST DCI (1792) 3-28 (1796/99) F-40  
 Test Equipment Commands, 3-1, A-1  
 Test Equipment Reserved Words, 2-3  
 TEST FREQ, 3-31  
 TEST LGC, 3-5, 3-7  
 TEST Monitor, 1-2  
 Test Numbers, 2-8, 3-14  
 TEST PSI, 3-25  
 Test Psuedo-Op, B-11  
 TEST PSV, 3-26

Test-Register Correlation, B-9  
 TEST DCV (1792) 3-27 (1796/99) F-40  
 Text Data Item, B-8  
 TGL (CAPS), C-8  
 TH, 3-2, 3-5, C-7  
 TIME, 2-8  
 Time Measurement Unit,(TMU) 1-4, 3-35, F-56  
 Time FRATIO Register, 3-31  
 TIN (FMU), 3-30  
 TINA (FMU), 3-30  
 TINB (FMU), 3-30  
 TL, 3-2, 3-5, C-7  
 TMU, 1-4, 2-8, 3-35  
 TPS, 1-5, D-2, (1796/99) F-55  
 TR, 2-7, 3-30  
 Translator, 1-2, 2-1  
 TRI 2-7, 3-29  
 TRIG 2-7, 3-29  
 TRIGA 2-7, 3-30  
 TRIGB 2-7, 3-30  
 Trigger Pulses, 3-26  
 Trigonometric Operations, 3-42  
 Triple Power Supply, 1-5, D-2, (1796/99) F-55  
 TS 2-7, 3-33  
 TTY, 1-3, 3-48, 3-51, B-10  
 TX, 3-2, 3-5, C-7  
 TYPE=n Probe (1796), F-43

- U -

Universal Power Supply, 1-4, 3-22, 4-1, D-3, F-53  
 UPS, 1-4, 3-22, 4-1, D-3, F-53  
 UPS Power Supplies (1792) 3-22, (1799) D-3, (1796/99) F-53  
 URDEFS=1, B-3, B-5  
 URFLD=INTEGER, B-3  
 USER 2-7, F-7  
 User Option Registers, 2-8, C-5  
 User Option Switches (USOPT1, USOPT2), 3-54  
 User-Styled Macros, 3-63  
 USOPT1 and USOPT2, 2-8, 3-54, C-5  
 USRPTN, B-4  
 Utility Commands, 3-1, 3-42, A-5  
 UUT, 1-1

- V -

VCG 2-7, 3-29  
 VD 2-7, 3-33  
 VH, 3-2, C-7  
 VIH, 1-3  
 VIHA=, 3-5  
 VIHb=, 3-5  
 VIL, 1-3  
 VILA=, 3-5  
 VILb=, 3-5  
 VF 2-7, 3-35  
 VL, 3-2, C-7  
 VL Mode, 3-22  
 VM 2-7, F-21  
 VO=OPTION, 3-19, 3-20, 3-22  
 VOH, 1-3  
 VOHA=, 3-5  
 VOHB=, 3-5  
 VOL, 1-3  
 VOLA=, 3-5  
 VOLB=, 3-5  
 Voltage Level Programming, 1-3  
 Voltage Locked (VL) Mode, 3-22  
 VP 2-7, F-21  
 VS 2-7, 3-33

# INDEX (CONT'D)

- W -

WAIT=n (1796) F-7  
Wrap-Around, 3-60  
WRITE DEV = Command, 3-48  
Writing Commands, 3-47  
WRITX, B-10

- X -

XMEAS (1792) 3-38,  
(1796/99) F-28  
Xmit Key, 3-51  
XSOURCE (1792) 3-38,  
(1796/99) F-28

- Z -

Z (1796) 2-7, F-6  
ZBRAND, B-6  
ZERO (1796/99), F-36, F-41  
Zero-Page Links, B-1, B-3  
ZGETFP, B-8  
ZGETIN, B-8  
ZGTTP, B-8  
ZHDERO, B-6  
ZPAUSE, B-7  
ZREAD, B-10  
ZTEX, B-11  
ZWRITE, B-11



# GenRad

300 Baker Avenue  
Concord, Massachusetts 01742

## ATLANTA

Alabama, Florida, Georgia, Mississippi, North Carolina,  
South Carolina, Tennessee

1 Dunwoody Park, Suite 107, Atlanta, GA 30341  
Tel: 404/394-5380

## \*BOSTON

Connecticut, Maine, Massachusetts, New Hampshire,  
Rhode Island, Vermont

### (SALES)

Route 117, Bolton, MA 01740  
Tel: 617/646-0550

### (SERVICE)

300 Baker Avenue, Concord, MA 01742  
Tel: 617/369-8770 • TWX: 710 347-1051

## \*CHICAGO

Illinois, Iowa, Kansas, Minnesota, Missouri, Nebraska,  
North Dakota, South Dakota, Wisconsin

1067 E. State Parkway, Schaumburg, IL 60195  
Tel: 312/884-6900 • TWX: 910 291-1209

## DALLAS

Arkansas, Louisiana, Oklahoma, Texas

777 South Central Expressway, Suite 4-A,  
Richardson, TX 75080  
Tel: 214/234-3357 • TWX: 910 867-4771

## DAYTON

Kentucky, Indiana, Michigan, Ohio, Western Pennsylvania

3300 South Dixie Drive, Dayton, OH 45439  
Tel: 513/294-1500 • TWX: 810 459-1785

## \*LOS ANGELES

Arizona, Hawaii, Nevada (Clark County only),  
Southern California

P.O. Box 19500, 17361 Armstrong Avenue  
Irvine Industrial Complex, Irvine, CA 92714  
Tel: 714/540-9830 • TWX: 910 595-1762

## \*NEW YORK

Delaware, New Jersey, New York, Eastern Pennsylvania

380 Midland Avenue, Saddle Brook, NJ 07662  
Tel: (NJ) 201/791-8990, (NY) 212/964-2722  
TWX: 710 988-2205

## SAN FRANCISCO

Colorado, Idaho, Montana, Nevada (except Clark County),  
New Mexico, Northern California, Oregon, Washington,  
Wyoming, Utah

2855 Bowers Avenue, Santa Clara, CA 95051  
Tel: 408/985-0662 • TWX: 910 338-0291

Alaska 907/279-5741

## \*WASHINGTON, DC

Maryland, Virginia, West Virginia, (Washington, DC)

15 Firstfield Road, Gaithersburg, MD 20760  
Tel: 301/948-7071 • TWX: 710 828-9741

## CANADA

### \*GenRad, Ltd.

307 Evans Avenue, Toronto, Ontario M8Z 1K2  
Tel: 416/252-3395 • TELEX: 06-967624

Montreal 514/747-1052  
Ottawa Zenith 88630

## EUROPE, AFRICA, and NEAR EAST

### \*GenRad, Ltd.

Bourne End, Bucks SL8 5 AT, England  
Tel: (06285) 2 66 11 • TELEX: 851-848321

\*Paris (01) 797 07 39

\*Milano (02) 20 92 57

\*München (089) 40 18 01

\*Zürich (01) 55 24 20

## \*ASIA, PACIFIC, and LATIN AMERICA

GenRad, Marketing & International Division

300 Baker Avenue, Concord, MA 01742  
TWX: 710 347-1051 • TELEX: 92-3354  
Cable: GENRADCO CONCORD (Mass)



